On Computationalism. The Evolution of Scientific and Technical Ideas[1]

ALBEANU, Grigore
Spiru Haret University
Scientific Research Center in Mathematics and Computer Science
g.albeanu.mi@spiruharet.ro

Abstract

Computationalism as a way of comprehending knowledge through computing has gone through important stages in the last century and has returned to the present with new computational models and ideas on practical realizability. The paper presents the main directions of evolution of computationalism and the theoretical and practical models elaborated to date.

Keywords: *computationalism, neural networks, digital and analog computers, unconventional computers.*

ACM Classification: K.2

1. ***Introduction***

Since antiquity, philosophers have been interested in the brain and its functioning. However, the first formulation of the principles of core learning and associative memory was made in 1890 by American psychologist William James (A. Dumitras, 1997). Understanding how the human brain functioned was an important research topic, not only for philosophers and psychologists, but especially for cybernetics.

Of the ranks of cybernetics, the most valuable ideas have been produced by artists in Artificial Intelligence. Thus, even before the emergence of the first digital computer, computationalism was proposed as a way of explaining the cognitive phenomenon by McCulloch and Pitts (1943). Along with McCulloch and Pitts, other scholars have participated as well in the development of ideas and the removal of suspicions. We mention Julian Bigelow (computer engineer: 1913-2003), Arturo Rosenblueth (cybernetics: 1900-1970), Norbert Wiener (founder of cybernetics: 1894-1964), John von Neumann (great mathematician: 1903-1957), Edward Fredkin (pioneer in digital technologies: 1934-), Stephen Wolfram (physicist: 1959-), Gregory John Chaitin (mathematician-informatician: 1947-).

[1]Citation: Albeanu G., *On Computationalism. The Evolution of Scientific and Technical Ideas*, "An. Univ. Spiru Haret, Ser. Mat.-Inform.", 12(1), pp. 45-50, 2016.

Konrad Zuse (1967) was the first to expand computationalism across the universe, and Fredkin, Wolfram and Chaitin considered that "any process, any change that takes place in the universe, can be considered a computation."

In the following, we will show how researchers define computationalism as a tool of computational knowledge, we will review both theoretical proposed computational models and real or virtual instruments proposed as technical solutions. Finally, we will discuss the program of pancomputationalism.

2. ***Computationalism and computational mechanisms***

Understanding human intelligence with the help of calculus (especially the symbolic) was the main objective of research in the field of Artificial Intelligence. The first computers were considered true "electronic brains" because they were able to perform very complex arithmetic and logic operations, played chess, wrote poetry, that is, they performed operations that are usually performed only by man. In addition, due to the increasing execution speed and the expansion of the applicative software portfolio, it was believed that these machines would overcome the human capacities of creation and knowledge, making man a slave to these machines. These fears have not come true, and more and more research projects deal with computationalism-based topics and support ideas that knowledge is computing and that the brain learns by using a specific algorithm.

In computationalism, a distinction is made between calculability, computer and computing. Calculability highlights those classes of functions that can not be calculated using the Turing machine. A Turing machine is an abstract device with infinite memory, a read-write head of symbols (stored in the machine's memory), a status register (an initial state is given and the number of possible machine states is finite) and a program (action table of finite size). If the action table has at most one entry for each symbol-state combination, then the machine is a deterministic Turing machine.

If the action table contains multiple inputs for at least one state-symbol combination, then the computing mechanism is a non-deterministic Turing machine. However, the two types of machines are computationally equivalent (have the same computing power). Since we can encode the action table of any Turing machine in a string it follows that we can build a Turing T machine that takes from its memory a string describing a T1 machine action table followed by a string describing the T1 machine memory contents, and then calculates the output that the Turing T1 so encoded machine would calculate. The process continues for other entries with the same structure. According to Turing (1947), "it can be shown that a single special machine of this type can be done to do the work of all the others. [...] The special machine can be called a universal machine" [19].

From computability point of view, a universal Turing machine is Turing-complete, that is, can compute any recursive function, decide any recursive language, and accept any enumerable recursive language. According to the Church-Turing conjecture, the problems solved by a universal Turing machine are precisely those problems solvable by an algorithm or an effective calculation method for any reasonable definition of these terms.

Several theoretical models for the notion of algorithm have been proposed in the literature, but their equivalence has been shown. There are functions that are not computable. For example, the question of whether a particular Turing machine stops at a particular entry or at any entry, a problem known as the stoppage problem, has shown its undecidability in Turing's original work. The passage from von Neuman's sequential machine to machines with unconventional architectures only resulted in an increase in processing speed, but the class of problem-solving was not widened.

The computer (without reference to its type and architecture) is the physical realization of a machine specialized in calculations. Initially, the computers followed the von Neuman architecture being serial and executing, at one time, a single operation. The first computing mechanism called "computer" by his inventor was ABS (Atanasoff-Berry Computer, 1942) and was able to solve linear equation systems with sizes up to 29.

In the context of this paper, the interest in the computational mechanisms is only of a functional nature, and by the computational process we mean the generation of a string of symbols starting from a series of given symbols and from a lot of processing rules. From a practical point of view, there are three classes of symbol strings: data (the operands of an operation), results and tasks (specify the operation to be executed). A list of instructions is a program.

The computing mechanism is modularly structured from components such as: processing units (called processors), memory drives, input devices, and output devices. In addition, the calculation mechanisms are also classified in terms of the technical solution (mechanical, electromechanical, electronic, etc.) of the qualitative attributes (programmability, size, calculation speed, cost, etc.).

Next, we will consider only programmable computing mechanisms. Programmability can be hardware and / or software. If hardware programmability requires knowledge and skills specific to the practical implementation solution, software programmability requires knowledge of decision-making methods and at least one programming language.

Software programmability can be external (the program is loaded from an input environment being available on perforated tape or cards), or internal (there is a mechanism by which the program is copied, stored and released for execution). Programmable systems can be dedicated (ABC, hardware programmable) or general purpose. Current systems are universal computational mechanisms with limited possibilities only by the available memory. They use memory virtualization and allow for complex notations and operations. An example of complex notation is the high-level programming language (Java, C ++, Python, etc.).

Computational mechanisms based on symbolic computing are called digital systems, the performance depending on the coding (representation) methods. A special class of computational mechanisms is represented by analog computers. Analog systems are continuous, while digital computational mechanisms are discrete. Of course, any continuous system can be approximated, with some precision, with a discrete system, which makes this distinction not to explain the difference between the two classes of systems.

Differentiation can be best explained from a functional point of view. Inputs and outputs of analog systems differ essentially as real variables associated with physical quantities with real values within a bounded range whose variability is continuous in relation to time. The operations performed by digital systems are oriented on symbol strings, while analog system operations act on real variables. Analog systems act continuously on inputs, while discrete systems perform operations at certain times taking into account the system clock and the duration of the instruction to execute. Obtaining results in digital processing is the effect of executing a program, while for analog systems, is necessary to solve a system of differential equations.

There are differences in the accuracy of the calculation, with digital systems proving superior and robust. In addition, it is easy to explain the non-universal nature of analogue computational mechanisms.

In natural (biological) systems, learning leads to a change in classical conditional behavior (Pavlov) or instrumental (Thorndike). Maren, Harston and Pap, according to A. Dumitras, 1997, p. 48, say that learning is a smart process, and intelligent system is one that has "the ability to be aware and to use the past and present innovations, for future decisions in an unobtrusive manner ", i.e. the ability to be aware and to learn: "Learning can be defined as a change in behavior due to the training process. Performance is assessed before and after the learning process – how much he learned."

3. ***Theoretical computational models and technical achievements***

Modern computationalism was formulated by Warren McCulloch in the 1930s. In 1943, McCulloch and his collaborator, Walter Pitts, published the first paper in the field. The two authors considered functional mental relations to be computational in the sense of Turing, and the calculations are performed by specific neural mechanisms, for which they have also developed mathematical techniques for the design of neural circuits. If W. James (1890) referred to a possible dependence of neuron activity on the sum of its input signal values, McCulloch and Pitts even developed an artificial model known as the "McCulloch-Pitts neuron" or "simple perceptron" to which they demonstrated the representability of any finite logical expression with the McCulloch-Pitts neural networks. Thus began the odyssey of connectionism, a term introduced by O. Hebb in 1949. In 1958, based on the retinal cell biology model, Frank Rosenblatt defined the "perceptron" – a network composed of a layer of many units (simple perceptron) and opened the path of research towards different learning methods (supervised, unsupervised).

The first model with supervised learning was proposed in 1960 by Bernard Widrow and Marcian Hoff, called ADALINE (ADAptive LINEar – linear relationship between entry and exit). With the publication of *Perceptron* in 1969 by Marvin Minski and Seymour Pappert, research in the field of artificial neural networks weakened in intensity because the authors mentioned, having shown the limitation of the computing power of existing networks, said that a possible extension to multilayer networks would be sterile. Although there were no major projects in the field, the statement of the two authors was contradicted in the following period. First, it is worth noting the contributions obtained independently and published in 1972 by Teuvo Kohonen (Finland) and James Anderson (American neurophysiologist) who first proposed an "associative memory" and the second one an "interactive memory" with *architecture, learning algorithms,* and almost identical *transfer functions.*

In the same period, Shunichi Amari expanded his studies to self-learning theory, and A. Henr Klopf proposed a "drive-reinforcement learning" biological insight by maximizing the efficiency of excitatory synapses, respectively minimizing the effectiveness of inhibitory synapses. Almost 10 years later, the Hopfield (1982) model was also proposed, which also enjoyed a hardware implementation (1984) by AT & T Bell Laboratories. For the Hopfield model, the concept of neural network energy was introduced. The true revival of interest in artificial neural networks occurred in 1986, just after the publication of the book "Parallel Distributed Processing", written by James McClelland, David E. Rumelhart and the PDP research group. It was also proposed the reverse-propagation multilayer perceptron model by Paul Werbos and Dave Parker, applicable to the prediction of economic data. Since 1988, as soon as the DARPA project has been funded, for the field of artificial neural networks, new projects and applications, as well as university courses, have begun to emerge.

After 1980, in the era called neo-connectivism by J.D. Cowan and D.H. Sharp, new architectures have been created, new learning algorithms have been developed, and the range of applications has been expanded. Practically, artificial neural networks are currently used to recognize forms in general, character recognition, signal processing in general, prediction-optimization, noise elimination, data compression, image processing, vocal signal processing, diagnosis, control, knowledge processing, experiments psychology and virtual reality. In terms of hardware achievements (electronic, optical, hybrid), we can list: integrated circuits, coprocessors, processors, highly parallel computers. Thus, artificial neural networks have been called in many ways, so that several domains are disputing supremacy in pancomputationalism: parallel and distributed processing patterns, connection patterns, adaptive systems, self-organizing systems, neurocomputing, neuromorphic systems.

Pancomputationalism adds new computational models: fuzzy cybernetic systems, neuro-fuzzy hybrid systems, DNA computation, membrane computation, quantum calculus, etc.

4. ***Conclusions***

In this paper a study on computationalism was initiated. The main theoretical and practical models were analyzed. Because the evolution of computational systems is very dynamic, this material is to be expanded with models of pan-computationalism and to highlight current limits and possible future paths.

References

1. G. Albeanu, *Algoritmi şi limbaje de programare (Algorithms and Programming Languages)*, Editura Fundaţiei România de Mâine, Bucharest, 2001.
2. P.M. Asaro, *On the Origins of the Synthetic Mind: Working Models, Mechanisms, and Simulations*, PhD Thesis, Urbana, Ilinois, 2006.
3. M. H. Bickhard, *Troubles with Computationalism.* In R. Kitchener, W. O' Donohue (editori) "Psychology and Philosophy: Interdisciplinary Problems and Responses", 1996: 173-183.
4. R. Cordeschi, M. Frixione, *Computationalism under attack.* In M. Marrafa, M. De Caro, F. Ferretti (editori): "Cartographies of the Mind", Springer, 2007: 37-49.
5. D. Davenport, *Computationalism: The Very Idea*, "Conceptus" 14 (2000): 121-137.
6. Gordana Dodig-Crnkovic, V. C. Muller, *A Dialogue Concerning Two World Systems: Info-computational vs. Mechanistic.* In Mark Burgin, Gordana Dodig-Crnkovic (editori) Information and Computation, "World Scientific", 2009.
7. A. Dumitras, *Proiectarea reţelelor neurale artificiale*, Casa Editorială Odeon, Bucureşti, 1997.
8. N. Fresco, *On the Need to Better Understand Our Computers*, The University of New South Wales, Sydney, 2007: http://philsci-archive.pitt.edu/archive/00003777/01/The_need_to_better_understand_computation_AAP_2007.pdf
9. G. LaForte, P. J. Hayes, K. M. Ford, *Why Godel's Theorem Cannot Refute Computationalism*, "Artificial Intelligence" 104(1998): 265-286
10. P. S. Novikov, *Elemente de logică matematică*, Editura Ştiinţifică, Bucureşti, 1966.
11. B. Marchal, *Computation, Consciousness and the Quantum*, Brussels University, 2004.
12. G. Piccinini, *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" 141 (2004): 175-215.
13. G. Piccinini, A. Scarantino, *Computation versus Information Processing: Why Their Difference Matters to Cognitive Science*, "Studies in the History and Pilosophy of Science", Elsevier, 2008.
14. G. Piccinini, *The Resilience of Computationalism*, Philosophy of Science: http://cogprints.org/6834/1/The_Resilience_of_Computationalism.pdf
15. G. Piccinini, *Functionalism, computationalism, and mental states*, "Stud. Hist. Phil. Sci." 35(2004): 811-833.
16. G. Piccinini, *Computations and Computers in the Science of Mind and Brain*, PhD Thesis, University of Pittsburgh, 2003.
17. I. Văduva, G. Albeanu, *Introducere în modelarea fuzzy*, Editura Universităţii din Bucureşti, 2004.
18. R. A. Wilson, C. F. Craver, *Realization.* In P. Thagard (editor) "Handbook of the Philosophy of Psychology and Cognitive Science", Kluwer, 2006.
19. ***, http://www.rutherfordjournal.org/article040101.html