The Nature of Computation and The Development of Computational Models

Mark Burgin\(^1\) and Gordana Dodig-Crnkovic\(^2\)

\(^1\) Department of Mathematics, UCLA, Los Angeles, USA
mburgin@math.ucla.edu,

\(^2\) School of Innovation, Design and Engineering, Mälardalen University, Västerås, Sweden
gordana.dodig-crnkovic@mdh.se

Abstract. We need much better understanding of information processing and its primary form computation than we have now. As there is no information without (physical) representation, the dynamics of information is implemented on different levels of granularity by different physical processes, including the levels of computation performed by computing machines and living organisms. There are a lot of open problems related to the nature of information and essence of computation, as well as to their relationships. How is information dynamics represented in computational systems, in machines, as well as in living organisms? Are computers processing only data or information and knowledge as well? What do we know of computational processes in machines and living organisms and how these processes are related? What can we learn from natural computational processes that can be useful for information systems and knowledge management?

1 Introduction

Many researchers have asked the question "What is computation?" trying to find a universal definition of computation or, at least, a plausible description of this important type of processes (cf. for example [1] [2] [3] [4] [5] [6]).

Some did this in an informal setting based on computational and research practice, as well as on philosophical and methodological considerations. Others strived to build exact mathematical models to comprehensively describe computation, and when Turing machine was constructed and accepted as a universal computational model, they imagined achieving the complete and exact definition of computation. However, the absolute nature of a Turing machine was disproved and in spite of all efforts, the conception of computation remains too vague and ambiguous.

This vagueness of foundations has resulted in a variety of approaches, including approaches that contradict each other. For instance, [3] writes "to compute is to execute an algorithm." Active proponents of the Church-Turing Thesis, such as [7] claim computation is bounded by what Turing machines are doing. For them the problem of defining computation was solved long ago with the Turing machine model. On the other hand, Wegner and Goldin insist that computation is an essentially broader concept than algorithm [8] and propose interactive view of computing. At the same time [9] argues
that computation is symbol manipulation. Neuroscientists on the contrary describe sub-symbolic computation in neurons. [10]

Existence of various types and kinds of computation, as well as a variety of approaches to the concept of computation, shows the complexity of understanding of computation in the holistic picture of the world. To work out the situation, we analyzed processes of concept clarification in science and mathematics when attempts were made at finding comprehensive definitions of basic scientific and mathematical ideas.

For instance, mathematicians tried to define a number for millennia. However, all the time new kinds of numbers were introduced changing the comprehension of what a number is. Looking back we see that at the beginning, numbers came from counting and there was only a finite amount of numbers. Then mathematicians found a way to figure out the infinite set of natural numbers, constructing it with 1 as the building block and using addition as the construction operation. As 1 played a specific role in this process, for a while, mathematicians excluded 1 from the set of numbers. At the same time, mathematicians introduced fractions as a kind of numbers. Later they understood that fractions are not numbers but only representations of numbers. They called such numbers rational as they represented a rational, that is, mathematical, approach to quantitative depiction of parts of the whole. Then a number zero was discovered. Later mathematicians constructed negative numbers, integer numbers, real numbers, imaginary numbers and complex numbers. It looked like as if all kinds of numbers had been already found. However, the rigorous representation of complex numbers as vectors in a plane gave birth to diverse number-like mathematical systems and objects, such as quaternions, octanions, etc. Even now only few mathematicians regard these objects as numbers.

A little bit later, the great mathematician Cantor [11] introduced transfinite numbers, which included cardinal and ordinal numbers. So, the family of numbers was augmented by an essentially new type of numbers and this was not the end. In the 20th century, [12] introduced nonstandard numbers, which included hyperreal and hypercomplex numbers. Later [13] founded surreal numbers and [14] established hypernumbers, which included real and complex hypernumbers. This process shows that it would be inefficient to restrict the concept of a number by the current situation in mathematics. This history helps us also to come to the conclusion that it would be unproductive to restrict the concept of computation by the current situation in computer science and information theory.

In this paper, we present historical analysis of the conception of computation before and after electronic computers were built and computer science emerged, demonstrating that history brings us to the conclusion that efforts in building such definitions by traditional approaches would be inefficient, while an effective methodology is to find essential features of computation with the goal to explicate its nature and to build adequate models for research and technology.

Consequently, we study computation in the historical perspective, demonstrating the development of this concept on the practical level related to operations performed by people and computing devices, as well as on the theoretical level where computation is represented by abstract (mostly mathematical) models and processes. This allows
us to discover basic structures inherent for computation and to develop a multifaceted typology of computations.

The paper is organized in the following way. In Section 2, we study the structural context of computation, explicating the Computational Triad and the Shadow Computational Triad. In Section 3, we develop computational typology, which allows us to extract basic characteristics of computation and separate fundamental computational types. The suggested system of classes allows us to reflect a natural structure in the set of computational processes. In Section 4 we present the development of computational models, and particularly natural computing. Finally, we summarize our findings in Section 5.

2 Structural Context of Computation

The first intrinsic structure, the Computational Dyad was introduced in [15], (Figure 1):

![The Computational Dyad](image)

Fig. 1: The Computational Dyad

The Computational Dyad reflects the existing duality between computations and algorithms. According to [5], in the 1970s Dijkstra defined an algorithm as a static description of computation, which is a dynamic state sequence evoked from a machine by the algorithm. Later a more systemic explication of the duality between computations and algorithms was elaborated. Namely, computation is a process of information transformation, which is organized and controlled by an algorithm, while an algorithm is a system of rules for a computation [4]. In this context, an algorithm is a compressed informational/structural representation of a process.

Note that a computer program is an algorithm written in (represented by) a programming language. This shows that an algorithm is an abstract structure and it is possible to realize one algorithm as different programs (in different programming languages). Moreover, many people think that neural networks perform computations without algorithms. However, this is not true because neural networks algorithms have representations that are very different from traditional representations of algorithms as systems of rules/instructions. The neural networks algorithms are represented by neuron weights and connections between neurons. This is similar to hardware representation Realization of algorithms in computers (analog computing). However, the Computational Dyad is incomplete because there is always a system that uses algorithms to organize and control computation. This observation shows that the Computational Dyad has to be extended to the Computational Triad (cf. Figure 2).
Note that the computing device can be either a physical device, such as a computer, or an abstract device, such as a Turing machine, or a programmed (virtual or simulated) device when a program simulates some physical or abstract device. For instance, neural networks and Turing machines are usually simulated by programs in conventional computers. Or Java virtual machine can be run on different operating systems and is processor and operating system independent.

Besides, with respect to architecture, it can be an embracing device, in which computation is embodied and exists as a process, or an external device, which organize and control computation as an external process. It is also important to understand the difference between algorithm and its representation or embodiment. An algorithm is an abstract structure, which can be represented in a multiplicity of ways: as a computer program, a control schema, a graph, a system of cell states in the memory of a computer, a mathematical system, such as an abstract finite automaton, etc.

In addition, there are other objects essentially related to computation. Computation always goes in some environment and within some context. Computation always works with data performing data transformations. Besides, it is possible to assume that computation performs some function and has some goal (for some agent) even if we don’t know this goal. The basic function of computation is information processing. These considerations bring us to the Shadow Computational Triad (cf. Figure 3).

Thus, the Shadow Computational Triad complements the Computational Triad reflecting that any computation has a goal, goes on in some context, which includes environment, and works with data. In a computation, information is processed by data transformations.

3 Computational Typology

There are many types and kinds of computations utilized by people and known to people. The structure of the world [16] implies the following classification.
3.1 Existential/substantial typology of computations
1. Physical or embodied computations.
2. Abstract or structural computations.
3. Mental or impersonalized computations.

According to contemporary science, abstract and mental computations are always represented by some embodied computations. The existential types from this typology have definite subtypes. There are three known types of physical/embodied computations.

1.1. Technical computations.
1.2. Biological computations.
1.3. Chemical computations.

Researchers discern three types of structural/abstract computations.
2.1 Symbolic computations.
2.2 Subsymbolic computations.
2.3 Iconic computations.

There are connections between these types. For instance, as [17] suggests, the principle of object formation may be an example of the transition from a stream of massively parallel subsymbolic microfunctional events to symbol-type, serial processing through subsymbolic integration. In addition to the existential typology, there are other typologies of computations.

3.2 Spatial typology of computations
1. Centralized computations where computation goes controlled by a single algorithm.
2. Distributed computations where there are separate algorithms that control computation in some neighbourhood. Usually a neighbourhood is represented by a node in the computational network.

3. Clusterized computations where there are separate algorithms that control computation in clusters of neighborhoods.

Turing machines, partial recursive functions and limit Turing machines are models of centralized computations. Neural networks, Petri nets and cellular automata are models of distributed computations. Grid automata in which some nodes represent networks with the centralized control [4] and the World Wide Web are systems that perform clusterized computations.

3.3 Temporal typology of computations
1. Sequential computations, which are performed in linear time.
2. Parallel or branching computations, in which separate steps (operations) are synchronized in time.
3. Concurrent computations, which do not demand synchronization in time.

Note that while parallel computation is completely synchronized, branching computation is not completely synchronized because separate branches acquire their own time and become synchronized only in interactions.

3.4 Representational or operational typology of computations
1. Discrete computations, which include interval computations.
2. Continuous computations, which include fuzzy continuous computations.
3. Mixed computations, some include are discrete and continuous processes.

Digital computer devices and the majority of computational models, such as finite automata, Turing machines, recursive functions, inductive Turing machines, and cellular automata, perform discrete computations. Examples of continuous computations are given by abstract models, such as general dynamical systems [18] and hybrid systems [19], and special computing devices, such as the differential analyzer [20] [21]. Mixed computations include piecewise continuous computations, combining both discrete computation and continuous computation. Examples of mixed computations are given by neural networks [22], finite dimensional machines and general machines of [23].

3.5 Hierarchy of levels of computation
In [6] three generality levels of computations are introduced.
1. On the top and most abstract/general level, computation is perceived as any transformation of information and/or information representation. 2. On the middle level, computation is distinguished as a discretized process of transformation of information and/or information representation. 3. On the bottom, least general level, computation is recognized as a discretized process of symbolic transformation of information and/or symbolic information representation.

There are also spatial levels or scales of computations:
1. The macrolevel includes computations performed by electromechanical devices, devices based on vacuum tubes and/or transistors, as well as mechanical calculators. 2. The microlevel includes computations performed by integrated circuits. 3. The nanolevel includes computations performed by fundamental parts that are not bigger than a few
nanometers. 4. The molecular level includes computations performed by molecules. 5. The quantum level includes computations performed by atoms and sub-atomic particles.

There are no commercially available nanocomputers, molecular or quantum computers in existence at present. However, current chips produced by nanolithography are close to computing nanodevices because their basic elements are less than 100 nanometers in scale.

4 Natural Computing

The development of computing, both machinery and its models, continues. We are used to quick increase of computational power, memory and usability of our computers, but the limit of miniaturization is approaching as we are getting close to quantum dimensions of hardware. One of the ideals of computing ever since time of Turing is intelligent computing which would besides mechanical include even intelligent problem solving. Currently, there is a development of cognitive computing aimed towards human abilities to process/organize/understand information. At the same time development of computational modelling of human brain has for a goal to reveal the exact mechanisms of human brain function that will help us understand not only how humans actually perform symbol processing when they follow an algorithm, but also how humans create algorithms or models. Those new developments 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 [24] natural computing is “the field of research that investigates both human-designed computing inspired by nature and computing taking place in nature.” In particular, it addresses: computational models inspired by the natural systems, computation performed by natural materials and computational nature of processes taking place in (living) nature. It includes among others areas of cellular automata and neural computation, evolutionary computation, molecular computation and quantum computation and 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 the natural sciences are rapidly absorbing ideas, tools and methodologies of information processing, computer science is broadening the notion of computation, recognizing information processing found in nature as natural computation. [25] [26] [27] That led Denning [28] to argue that computing today is a natural science.

This new concept of computation with inspiration in natural information processing allows among others learning about nondeterministic complex computational systems with self-* properties (self-organization, self-configuration, self-optimization, self-healing, self-protection, self-explanation, and self-awareness. Natural computation thus provides a basis for a unified understanding of phenomena of embodied cognition, intelligence and knowledge generation. [29] [30] In the development of unconventional algorithms [31]nemphasize the complementarity of axiomatics and construction, both elements being necessary for the progress in our understanding of computation. New powerful tools are brought forth by local mathematics, local logics, logical varieties
and the axiomatic theory of algorithms, automata and computation. Further work includes study of natural computation by unconventional algorithms and constructive approaches.

5 Conclusion

This paper presents a study in the nature of computation, contributing with computation typologies: essential, spatial, temporal, representational and hierarchy-level based. We also address the developments of models of computation, with emphasis on physical/embodied computation and concurrent models. Our analysis suggests that much better understanding of computation is needed than we have today. There are numerous open problems related to the nature of information and essence of computation, as well as to their relationships. How is information dynamics represented in computational systems, in machines, and in living organisms? What do we know of computational processes in machines, living organisms and in the physical world in general and how are these processes related? What can we learn from natural computational processes that can be useful for information systems, knowledge management and understanding of intelligence?

Present account of models of computation highlights several topics of importance for the development of new understanding of computing and its role: 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 modelling needed to support this generalized framework.

References