A Taxonomy of Computation and Information Architecture

Mark Burgin
Department of Mathematics,
UCLA, Los Angeles, USA
mburgin@math.ucla.edu

Gordana Dodig-Crnkovic
Chalmers University of Technology
and University of Gothenburg
dodig@chalmers.se

ABSTRACT

Nowadays computation is typically understood through the Turing machine model, in the fields of computability, computational complexity and even as a basis for present-day computer hardware and software architectures. Those are technologies designed in the first place to process data. Being description of data manipulation, Turing model of computation presents only one aspect of computation in the real world – an abstraction of the execution of an algorithm. However, several other possible aspects of computation, even those existing in today's applications, are left outside, thus adequate models in distributed, self-organized, resource-aware, adaptive, learning computation systems are needed. This paper presents taxonomy of existing models of computation. It is connected to more general notion of natural computation, intrinsic to physical systems, and particularly cognitive computation in cognitive systems. We see Turing model of computation as a basic mechanism which can be used to build more complex computational architectures, that in combination with interaction with the environment (learning) give advanced information-processing behaviors in cognitive systems.

Keywords

Computation, Information Processing, Algorithms, Turing Machine, Computing Architecture, Computing Taxonomy, Cognitive computing

1. INTRODUCTION

Future progress of new computational devices capable of dealing with problems of big data, internet of things, semantic web, cognitive robotics and neuroinformatics depends on the adequate models of computation. In this article we first present the current state of the art through systematization of existing models and mechanisms, and outline basic structural framework of computation. We argue that defining computation as information processing, and given that there is no information without (physical) representation, the dynamics of information on the fundamental level is physical/ intrinsic/ natural computation. As a special case, intrinsic computation is used for designed computation in computing machinery. Intrinsic natural computation occurs on variety of levels of physical processes, containing the levels of computation of living organisms (including highly intelligent animals) as well as designed computational devices.

The present article offers taxonomy of current models of computation and indicates future paths for the advancement of the field; both by the development of new computational models and by learning from nature how to better compute using different mechanisms of intrinsic computation.

The question "What is computation?" is answered differently by different researchers (cf., for example, [1]–[8]). 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. When the Turing machine (or Logical Computing Machine as Turing originally named his logical device) was constructed and accepted as a universal computational model, it was considered as the complete and exact definition of computation (Church-Turing thesis). However, the absolute nature of the Turing machine was disproved by adopting a more general definition of algorithm [4].

In the spirit of broadening of the concept of computation, the following definition was proposed: "For a process to qualify as computation a model must exist such as algorithm, network topology, physical process or in general any mechanism which ensures definability of its behavior." [9]

Nevertheless, in spite of all efforts, the conception of computation remains too vague and ambiguous. This vagueness of the foundations of computing 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 [10] claim computation is bounded by what Turing machines are doing (that is compute mathematical functions). For them the problem of defining computation was solved long ago with the Turing machine model. At the same time, Wegner and Goldin insist that computation is an essentially broader concept than algorithm [11] and propose interactive view of computing. While [12] argues that computation is symbol manipulation, thus disregarding analog computers computing over continuous signals, neuroscientists on the contrary study sub-symbolic computation in neurons. [13] Abramsky summarizes the process of successive changing of computing models as follows:

"Traditionally, the dynamics of computing systems, their unfolding behavior in space and time has been a mere means to the end of computing the function *which specifies the algorithmic problem which the system is solving. In much of contemporary computing, the situation is reversed: the purpose of the computing system is to exhibit certain* behaviour. *(...) We need a theory of the dynamics of informatic processes, of interaction, and information flow, as a basis for answering such fundamental questions as: What is computed? What is a process? What are the analogues to Turing completeness and universality when we are*

concerned with processes *and their* behaviors*,* rather than the functions which they compute? [14]

Abramsky emphasizes that there is the need for *second-generation models of computation*, and in particular process models. *The first generation models of computation* originated from problems of formalization of mathematics and logic, while processes or agents, interaction, and information flow are results of recent developments of computers and computing. In the second-generation models of computation, previous isolated systems are replaced by processes or agents for which the *interactions* with each other and with the environment are fundamental. Hewitt too advocates agent-type, Actor model of computation [8] which is suitable for modeling of physical (intrinsic) computation.

Existence of various types and kinds of computation, as well as a variety of approaches to the concept of computation, shows remarkable complexity that makes communication of results and ideas increasingly difficult. Our aim is to explicate present diversity and so call attention to the necessity of common understanding: different models of computation may have their specific uses and applications. It is just necessary to understand their mutual relationships and assumptions under which they apply.

In this paper, we present methodological analysis of the concept of computation before and after electronic computers and the emergence of computer science, demonstrating that history brings us to the conclusion that efforts in building such definitions by traditional approaches would be inefficient. 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. In addition, we perform structural analysis of the concept of computation through explication of various structures intrinsically related to computation, and in particular cognitive computation.

One important question related to computation (information processing) architecture is if cognition can be understood as computation in the sense of Turing-Church thesis or if cognitive computation needs additional constructs in case of self-managing (autonomous) distributed computing systems such as human brain. Cognition is taken to be the ability to process information, apply knowledge, and act accordingly, with intent, and it s accomplishment through various processes that monitor and control a system and its environment. "Cognition is associated with a sense of "self" (the observer) and the systems with which it interacts (the environment or the "observed"). Cognition extensively uses time and history in executing and regulating tasks that constitute a cognitive process. In [15], [16] Mikkilineni argues that "cognition requires more than mere book-keeping provided by the Turing machines and certain aspects of cognition such as self-identity, self-description, self-monitoring and self-management can be implemented using parallel extensions to current serial von-Neumann stored program control (SPC)."

The paper is organized in the following way. In Section 2 we develop computational taxonomy, which allows us to extract basic characteristics of computation and distinguish fundamental types of computation. The suggested system of computational classes allows us to reflect natural structures in the set of computational processes. In Section 3 we study the structural context and architecture of computation, illuminating the computational triad and several other structures intrinsically related to computation. In Section 4 we present the development of computational models in natural computing (computing nature). Finally, we summarize and present open questions in Section 5.

2. COMPUTATIONAL TAXONOMY

In his famous work [1] Turing presents his fundamental model, which later was called the Turing machine model of computation. Turing writes: "We may now construct a machine to do the work of this computer." Here a computer is a *person* performing mechanical procedure executing an algorithm. Thus, algorithms were the first models of computation. Algorithm was the practice of algebra in the 18th century. In the 19th century, the term came to mean any process of systematic calculation. In the 20th century, Encyclopedia Britannica described algorithm as a systematic mathematical procedure that produces – in a finite number of steps – the answer to a question or the solution of a problem.

It is common that we talk about computation as if it would be a uniquely defined concept. However some think of computation as algorithm, others as symbol manipulation, while yet others may have in mind a more general phenomenon of information processing.

Currently there are so many kinds of concepts of computation and its implementations that it is useful to make classification and systematization so to better understand their mutual relationships. In what follows we will present several main ttaxonomies of computation: existential/substantial, organizational, temporal, representational, data-oriented, operational, process-oriented and level-based.

2.1 Existential taxonomy

On the different levels of organisation of objects/agents (subjects) (physical, structural, interpretant) we find different types of computational processes. According to Burgin, [17] p. 93, reality in the most abstract way can ben represented by the existential triad (physical, structural, mental[1]) which corresponds to Plato's triad (material, ideas/forms, mental). This can also be related to the Peirce's triad of (object, sign, interpretant).

The existential/substantial classification of computation based on the existential triad [18] defines the following types:

1. *Physical or embodied (object)* computations
2. *Abstract or structural (sign)* computations
3. *Mental or cognitive (interpretant)* computations

The existential types from this taxonomy have definite subtypes. The following are types of embodied computations, where each next level emerges as a consequence of previous ones.

1.1. *Physical* computations

1.2. *Chemical* computations

1.3. *Biological* computations

1.4. *Cognitive* computations

[1] Here *mental* denotes that which is *of or relating to the mind. The* mind is a traditional and vague concept and in scientific terms it corresponds to an emergent process that emerges from cognition. Unlike *mind, which* is ascribed primarily to humans, *cognition* can be attributed to both animals and machines and is therefore a more useful concept in the context of computational architecture.

With respects to the structures (objects) that are processed by computations, it is possible to discern the following types:

2.1 *Subsymbolic* computations - data/signal processing

2.2 *Symbolic* computations - data structures processing

2.3 *Hybrid/mixed* subsymbolic and symbolic computations.

There are connections between the above types. For instance [19] differentiates between *verbal* (linguistic) (symbolic and subsymbolic) and non-verbal (non-linguistic) (symbolic and subsymbolic) mental (cognitive) processes. He suggests that the principle of object formation may be an example of the transition from a stream of massively parallel subsymbolic micro-functional events to symbol-type, serial processing through subsymbolic integration. Even [20] suggests similar connection between symbolic and subsymbolic (connectionist) computations.

Mental (cognitive, interpretive) computations can be observed at the following levels:

3.1 *Individual* (computational network of the brain)

3.2 *Group* (computational networks of individuals)

3.3 *Social* (computational networks of groups)

A taxonomy of computation in relation to cognition is provided by Fresco [21], who notes that, depending on the specific kind of "information" used, information-processing accounts of computation may differ greatly. Four main types of information in this context are given:

3.a *Syntactic* (Shannon) information

3.b *Algorithmic* information

3.c Semantic information

3.d *Instructional* information

2.2 Organizational taxonomy

Organization of computation can be characterized as:

1. *Centralized computations* where computation is controlled by a single algorithm.
2. *Distributed computations* where there are separate algorithms that control computation in some neighbourhood that is represented by a node in the computational network.
3. *Clustered computations* where there are separate algorithms that control computation in clusters of neighbourhoods.

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 and the World Wide Web are systems that perform clustered computations [4]. Grid automaton is the most advanced abstract model of distributed computing systems, which performs concurrent computations, while being a physical distributed computing system.

2.3 Temporal taxonomy

With respect to temporal characteristics, computations can be:

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.

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.

Classical models of computation, such as the classical Turing machine or partial recursive functions, perform only sequential computations. Models that appeared later, such as Turing machines with several heads and tapes or cellular automata, provide means for parallel computations. There are also various models for concurrent computations, according to [22].

In this context, the most advanced device model is grid automaton, while the most advanced operational model, which also is a process model, is the EAP (event-action-process) model. All these models form three classes:

3.1 *Device models* (Petri nets, Kahn process networks, dataflow process networks, discrete event simulators, grid automata, the Linda model and the Actors model).

3.2 *Operational models* (ACP (algebra of communicating processes), VCR (extended view-centric reasoning), EVCR (view-centric reasoning) and ESP (event-signal-process) model).

3.3 *Process models* (CSP (communicating sequential processes) model and CCS (composite current source delay) model.

2.4 Representational taxonomy

With respect to the data representation on which computations are performed, there are following types of computation:

1. *Discrete* computations, which include interval computations.
2. *Continuous* computations, which include fuzzy continuous computations.
3. Hybrid/*mixed* computations, which include discrete and continuous processes.

Digital computing 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 [23] and hybrid systems [24], and special computing devices, such as the differential analyzer [25][26].

Hybrid/Mixed computations include piecewise continuous computations, combining both discrete computation and continuous computation. Examples of mixed computations are

given by neural networks [27], finite dimensional machines and general machines of [28].

It is possible to refine the representational taxonomy in the following way, obtaining three additional classifications.

2.5 Data-based taxonomy

With respect to the data and the domain of computation, the following possibility exist:

1. The domain of computation is *discrete* and data are *finite*. For instance, data are words in some alphabet.
2. The domain of computation is *discrete* but data are *infinite*. For instance, data are ω-words in some alphabet. This includes interval computations because real numbers traditionally are represented as ω-words.
3. The domain of computation is *continuous*.

2.6 Operational taxonomy

With respect to operations in computation, the following taxonomy can be found:

1. Operations in computation are *discrete* and they transform *discrete* data elements. For instance, addition or multiplication of whole numbers.
2. Operations in computation are *discrete* but they transform (operate with) *continuous* sets. For instance, addition or multiplication of all real numbers or of real functions.
3. Operations in computation are *continuous*. For instance, integration of real functions.

2.7 Process-oriented taxonomy

1. The process of computation is *discrete*, i.e. it consists of separate steps in the *discrete* domain, and it transforms discrete data elements. For instance, computation of a Turing machine or a finite automaton.
2. The process of computation is *discrete* but it employs *continuous* operations. An example is given by analogue computations [25][26] as quoted in [4] p. 122.
3. The process of computation is *continuous* but it employs discrete operations. For instance, computation of a limit Turing machine [4] p. 140.

2.8 Computation levels taxonomy

In [6] three *generality levels of computations* are introduced:

1. At the top and most abstract/general level, computation is perceived as *any transformation of information* and/or information representation.
2. At the middle level, where computation is distinguished as a *discretized process* of transformation of information and/or information representation.
3. At the bottom, least general level, computation is defined as a *discretized process of symbolic transformation* of information and/or symbolic information representation.

There are also *spatial levels* or *scales* of computations. Even though the highest levels subsume the lower ones, computation is performed/interpreted at a given level, as follows:

1. Macro-level includes computations performed by mechanical calculators as well as electromechanical devices.
2. Micro-level includes computations performed by integrated circuits.
3. Nano-level includes computations performed by fundamental parts that are not bigger than a few nano meters.
4. Molecular level includes computations performed by molecules.
5. Quantum level includes computations performed by atoms and subatomic particles.

At present there are no commercially available nano-computers, molecular or quantum computers, but they are being developed.

3. STRUCTURAL FRAMEWORK AND ARCHITECTURE OF COMPUTATION

The earliest idea of computation was application of an algorithm. The Turing machine model of computation is equivalent to an algorithm. Thus, the first and most commonly encountered computational structure is the *computational dyad* [29] (computation, algorithm). The *computational dyad* reflects the existing duality between computations and algorithms. In 1971 Dijkstra in his *A Short Introduction to the Art of Programming* [30] defined an *algorithm as a static description of computation*, which is a dynamic state sequence induced in 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. A computer program is an algorithm written in (represented by) a programming language.

Thus, an algorithm is an abstract structure and it is possible to realize the same algorithm as different programs (in different programming languages). It is 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. Interestingly, many people think that neural networks perform computations without algorithms. However, this is not true as neural networks algorithms have representations even though 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).

The computational dyad is a simplification and incomplete because there is always a system that uses algorithms to organize and control computation. This observation suggests that the computational dyad has to be extended to the triangular basic computational triad (algorithm, computation, device/agent) that reflects that specifies computation is performed by a device, such as a computer, or by an agent, such as a human being or bacteria. Functioning of the device or work of the agent that/who performs computation is directed or controlled by an algorithm embodied in a program, plan, scenario or hardware. Consequently, the process of computation is also directed/controlled by an algorithm.

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 on conventional computers. A 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.

The *basic computational triad* reflects the structure of the world represented by the *existential triad* [17].

4. NATURAL COMPUTATION/ COMPUTING NATURE

In previous sections we presented concept of computation and its models and developed computational taxonomies in the context of information processing. Adopting the approach of structural realism we provided structural framework of computation based on Burgin's *existential triad* (physical, structural, mental/ cognitive) of the world [17]. We have chosen triadic structures as fundamental elements in our study of structures, based on the classical triad of Plato – (matter, structure, mind), Peirce's triad of (object, sign, interpretant) and Poppers structural triad (physical world, knowledge, mentality).

In what follows, we will concentrate on the first two elements of the existential triad – physical and structural in their cognitive aspect, both of them addressed through the idea of computing nature (natural computing), [7] [31].

According to the Handbook of Natural Computing [32] 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: intrinsic computation performed by natural materials and computational nature of processes taking place in nature. Natural computing comprises, among others, areas of cellular automata and neural computation, evolutionary computation, molecular, quantum and organic computing, biocomputing, nature-inspired algorithms.

Knowledge in natural computing is generated bi-directionally, through the interaction between computer science and natural sciences. As the natural sciences are promptly absorbing concepts, tools and methodologies of information processing, computer science on its side is broadening the notion of computation, recognizing information processing found in nature as special type of computation – intrinsic, natural computation. [33] [34][35][36] This development in understanding of computation led Denning to argue that computer science today is a natural science [37] [38].

In his "Simulating Physics with Computers" Feynman argued that the specific forms of computation are best carried out by the physical substrate we are trying to describe [39]. Even though DNA computation can be emulated *in silico*, the efficiency of computing with DNA substrate instead of its digital representation is higher by many orders of magnitude [40]. Similar view of physical (material) computation is presented in [41] where it is argued for clear benefits of intrinsic (material) computation, as in the Brook's famous observation that "the world is its own best model" [42]. Cooper addresses the question of the relationship between abstract mathematical models and physical computation that underlies every real-world computation, suggesting that we are under the spell of the "mathematician bias" and arguing for the return to embodied computation [43] [44]–[46]showing how nature can help us to compute.

4.1 Intrinsic vs. designed computation

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 components. We are also facing the explosive increase in the amounts of data, "big data" and the emergence of the "internet of things" where computers are becoming an integral part of practically all devices and indeed of our physical environment. All of this poses huge demands on effective computation techniques. Ever since the time of Turing, one of the ideals was intelligent computing, which would besides mechanical symbol manipulation include even intelligent problem solving. That would help us manage complexity and vast amounts of data that have to be processed, often in real time. In that direction currently, there is a development of cognitive computing [47]–[50] aimed towards human-level abilities of machines that process/organize/ and even understand information.

At the same time the development of computational models of human brain has for a goal to reveal the exact mechanisms of human brain function (https://www.humanbrainproject.eu) that will help us understand not only how humans actually perform information processing when they follow an algorithm, *but also how humans create algorithms or models in a recursive cascades of self-reflective computational processes from physical substrate to information/patterns to mental states through cognitive computing*. Those new developments can be seen as a part of the research within the field of natural computing, where natural system performing computation is embodied and embedded human brain.

The new understanding of computation as complex distributed concurrent computational system of systems with inspiration in natural information processing allows among others learning about nondeterministic complex computational systems, such as living organisms, with self-* properties (self-organization, self-configuration, self-optimization, self-healing, self-protection, self-explanation, and self-awareness). Natural computation has a potential to provide a basis for a unified understanding of phenomena of embodied cognition, intelligence and knowledge generation. [51][47]

Recently, a focus issue of the journal Chaos was dedicated to the intrinsic and designed computation under the title "Information Processing in Dynamical Systems—Beyond the Digital Hegemony" addressing challenges of intrinsic computing in dynamical systems as complementary to designed computing in digital systems. [36]

This relates to the view of a brain as a dynamical system processing information (computing) at different levels of organization – from molecular/electro-chemical, cellular processes (DNA, protein networks), through neural circuits, cortical through columns (morphologically distinct regions of the brain processing and exchanging information), and finally through the level of whole-brain information integration that is considered to provide the function of consciousness, [52]. Cellular and whole-brain levels of computation correspond to the cognition level of cells and the brain. In a biological sense, cognition is the property of an autopoietic system, with self-production, self-

organisation and closure and in structural coupling with the environment. [53]

At this point, there are several avenues of the future development of computing both in the setting of physical devices and of theoretical models. Natural computation promises new and broader ways of understanding of computational processes that can be found intrinsically in various physical systems. In connection to that, the structure and its particular case - architecture of computational processes becomes increasingly important, especially in the case of cognitive computing.

4.2 Levels of physical organization in natural computing as natural information processing

In the section on Taxonomy of computation, under the heading Hierarchy of computation levels, we mentioned levels and scales of physical organization at which computation is performed in designed computation. In the case of cognitive computing, the following levels of designed, nature-inspired computation are distinguished: synapses, neurons, microcircuits (modeling brains cortical columns), long range interconnections (modeling axons function) and the whole-brain level integration of processes, representing non-von-Neumann architecture:

"overarching cognitive computing architecture is an on-chip network of light-weight cores, creating a single integrated system of hardware and software. This architecture represents a critical shift away from traditional von Neumann computing to a potentially more power-efficient architecture that has no set programming, integrates memory with processor, and mimics the brain's event-driven, distributed and parallel processing." http://www-03.ibm.com/press/us/en/pressrelease/35251.wss

Mikkillineni [15], [16] describes cognitive architectures based on his Distributed Intelligent Computing Element (DIME) computing model, consisting of a "recursive managed distributed computing network, which can be viewed as an interconnected group of such specialized Oracle machines" that "provides the architectural resiliency, which is often associated with cellular organisms, through auto-failover; auto-scaling; live-migration; and end-to-end transaction security assurance in a distributed system." Mikkillineni argues that "the self-identity and self-management processes of a DIME network" add the elements of cognition into Turing machine type computing. This approach extends Burgin's view of computing consisting of hardware, software and infoware [4] with one more architectural layer corresponding to cognitive function related to knowledge.

5. CONCLUSIONS AND OPEN QUESTIONS

Present account suggests that the concept of computation as information processing develops together with the constantly increasing scientific knowledge and tools of analysis [54]. We present the structural framework of information processing and computation starting with existential triadic relationships between (physical, structural, mental/cognitive) and taxonomies of numerous aspects of computation that follow this basic existential layered architecture that is paralleled by physico-chemical, chemo-biological and bio-cognitive levels of information processing [55]. As there is no information without (physical) representation [18], the dynamics of information is implemented on different levels of granularity by different physical processes, including the level of computation performed by computing machines (designed computation), as well as by living organisms (cognitive computation).

We highlight several topics of importance for the development of new understanding of computation and its role: natural computation, interactivity as fundamental for computational modeling of concurrent information processing systems such as living organisms and their networks, and the new developments in modeling needed to support this generalized framework for cognitive architectures [4][16]. In such a way, we achieve better understanding of computation as information processing on different levels.

There are still many open problems related to the nature of information and 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 capable of processing only data or information and knowledge as well, as recent work of Mikkillineni suggests? What can we know of computational processes in machines and living organisms and how these processes are related to the computational architectures? What can we learn from natural computational processes in cognitive systems that can be useful for engineered information systems and knowledge management?

6. REFERENCES

[1] A. M. Turing, "On computable numbers, with an application to the Entscheidungs problem," *Proc. London Math. Soc.*, vol. 42, no. 42, pp. 230–265, 1936.

[2] A. N. Kolmogorov, "On the Concept of Algorithm," *Russ. Math. Surv.*, vol. 8, no. 4, pp. 175–176, 1953.

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

[4] M. Burgin, *Super-Recursive Algorithms*. New York: Springer-Verlag New York Inc., 2005.

[5] P. Denning, "What is computation?: Editor's Introduction," *Ubiquity*, no. October, pp. 1–2, 2010.

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

[8] C. Hewitt, "What is computation? Actor Model versus Turing's Model," in *A Computable Universe, Understanding Computation & Exploring Nature As Computation*, H. Zenil, Ed. World Scientific Publishing Company/Imperial College Press, 2012.

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

[10] L. Fortnow, “The enduring legacy of the Turing machine,” *Comput. J.*, vol. 55, no. 7, pp. 830–831, 2012.

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

[12] J. S. Conery, “Computation is symbol manipulation,” *Comput. J.*, vol. 55, no. 7, pp. 814–816, 2012.

[13] D. E. Angelaki, A. G. Shaikh, A. M. Green, and J. D. Dickman, “Neurons compute internal models of the physical laws of motion,” *Nature*, vol. 430, no. 6999, pp. 560–564, 2004.

[14] 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.

[15] R. Mikkilineni, *Designing a New Class of Distributed Systems*, SpringerBr. Berlin Heidelberg: Springer, 2011.

[16] R. Mikkilineni, “Going beyond Computation and Its Limits: Injecting Cognition into Computing,” *Appl. Math.*, vol. 3, pp. 1826–1835, 2012.

[17] M. Burgin, *Structural Reality*. New York: Nova Science Publishers, 2012.

[18] R. Landauer, “The Physical Nature of Information,” *Phys. Lett. A*, vol. 217, p. 188, 1996.

[19] W. Bucci, *Psychoanalysis and Cognitive Science: A Multiple Code Theory*. New York: Guilford Press, 1997.

[20] A. Clark, *Microcognition: Philosophy, Cognitive Science, and Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1989.

[21] N. Fresco, *Physical Computation and Cognitive Science*. Berlin Heidelberg: Springer Berlin Heidelberg, 2014.

[22] M. Burgin and M. L. Smith, “Concurrent Composition and Algebras of Events, Actions, and Processes,” *arXiv:0803.3099*, 2008.

[23] O. Bournez, “Achilles and the Tortoise climbing up the hyper-arithmetical hierarchy,” *Theor. Comput. Sci.*, vol. 210, pp. 210–211, 1977.

[24] V. Gupta, R. Jagadeesan, and V. A. Saraswat, “Computing with continuous change,” *Sci. Comput. Program.*, vol. 30, no. 1–2, pp. 3–49, 1998.

[25] C. Shannon, “Mathematical Theory of the Differential Analyzer,” *J. Math. Phys.*, vol. 20, pp. 337–354, 1941.

[26] C. Moore, “Recursion theory on the reals and continuous-time computation,” *Theor. Comput. Sci.*, vol. 162, no. 1, pp. 23–44, 1996.

[27] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity. 1943.,” *Bull. Math. Biol.*, vol. 52, no. 1–2, pp. 99–115; discussion 73–97, 1990.

[28] L. Blum, F. Cucker, M. Shub, and S. Smale, “Complexity and Real Computation: A Manifesto,” *Int. J. Bifurc. Chaos*, vol. 6, no. 1, pp. 3–26, 1996.

[29] M. Burgin and E. Eberbach, “Evolutionary Computation and Processes of Life,” *Ubiquity*, no. August, pp. 1–13, 2012.

[30] E. W. Dijkstra, “A Short Introduction to the Art of Computer Programming,” 1971.

[31] G. Dodig-Crnkovic and R. Giovagnoli, *Computing Nature*. Berlin Heidelberg: Springer, 2013.

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

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

[34] 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.

[35] 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.

[36] J. Crutchfield, W. Ditto, and S. Sinha, “Introduction to Focus Issue: Intrinsic and Designed Computation: Information Processing in Dynamical Systems—Beyond the Digital Hegemony,” *Chaos*, vol. 20, no. 037101, 2010.

[37] P. Denning, “Computing is a natural science,” *Commun. ACM*, vol. 50, no. 7, pp. 13–18, 2007.

[38] P. Denning and P. Rosenbloom, “The fourth great domain of science,” *ACM Commun.*, vol. 52, no. 9, pp. 27–29, 2009.

[39] R. P. Feynman, “Simulating Physics with Computers,” *Int. J. Theor. Phys.*, vol. 21, no. 6/7, pp. 467–488, 1982.

[40] M. Nadin, “Anticipatory Computing. From a High-Level Theory to Hybrid Computing Implementations,” *Int. J. Appl. Res. Inf. Technol. Comput.*, vol. 1, no. 1, pp. 1–27, 2010.

[41] S. Stepney, “The neglected pillar of material computation,” *Phys. D Nonlinear Phenom.*, vol. 237, no. 9, pp. 1157–1164, 2008.

[42] R. Brooks, “Elephants don’t play chess,” *Rob. Auton. Syst.*, vol. 6, pp. 3–15, 1990.

[43] 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.

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

[45] S. B. Cooper, “How Can Nature Help Us Compute?,” in *SOFSEM 2006: Theory and Practice of Computer Science - 32nd Conference on Current Trends in Theory and Practice of Computer Science*, 2006, pp. 1–13.

[46] S. B. Cooper, “The Mathematician’s Bias - and the Return to Embodied Computation,” in *A Computable Universe: Understanding and Exploring Nature as Computation*, H. Zenil, Ed. World Scientific Pub Co Inc, 2012.

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

[48] Y. Wang, “Toward a Formal Knowledge System Theory and Its Cognitive Informatics Foundations,” in *Transactions on Computational Science V*, vol. 5540, M. L. Gavrilova, C. J. K. Tan, Y. Wang, and K. C. C. Chan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–19.

[49] Y. Wang, “On Contemporary Denotational Mathematics for Computational Intelligence,” *Trans. Comput. Sci.*, vol. 2, pp. 6–29, 2008.

[50] Y. Wang, “The Theoretical Framework of Cognitive Informatics,” *Int’l J. Cogn. Informatics Nat. Intell.*, vol. 1, no. 1, pp. 1–27, 2007.

[51] G. Dodig-Crnkovic and V. Mueller, “A Dialogue Concerning Two World Systems: Info-Computational vs. Mechanistic,” World Scientific Pub Co Inc, Singapore, 2009.

[52] G. Tononi, “The Integrated Information Theory of Consciousness: An Updated Account,” *Arch. Ital. Biol.*, vol. 150, no. 2/3, pp. 290–326, 2012.

[53] H. Maturana, “Autopoiesis, Structural Coupling and Cognition: A history of these and other notions in the biology of cognition,” *Cybern. Hum. Knowing*, vol. 9, no. 3–4, pp. 5–34, 2002.

[54] G. Dodig-Crnkovic, “The Development of Models of Computation with Advances in Technology and Natural Sciences,” in *Symposium of The British Society for the Study of Artificial Intelligence, AISB 2013*, 2013.

[55] G. Dodig-Crnkovic, “Why we need info-computational constructivism,” *Constr. Found.*, vol. 9, no. 2, pp. 246–255, 2014.