proceedings

Proceedings

Modeling Computing Devices and Processes by Information Operators †

Mark Burgin [1,*] and Gordana Dodig-Crnkovic [2]

[1] Department of Computer Science, University of California, Los Angeles (UCLA), Los Angeles, CA 90095, USA
[2] Department of Computer Science and Engineering, Chalmers University of Technology, 41296 Gothenburg, Sweden; gordana.dodig-crnkovic@chalmers.se
* Correspondence: markburg@cs.ucla.edu

† Conference Theoretical Information Studies (TIS), Berkeley, CA, USA, 2–6 June 2019.

Published: 11 May 2020

Abstract: The concept of operator is exceedingly important in many areas as a tool of theoretical studies and practical applications. Here, we introduce the operator theory of computing, opening new opportunities for the exploration of computing devices, networks, and processes. In particular, the operator approach allows for the solving of many computing problems in a more general context of operating spaces. In addition, operator representation of computing devices and their networks allows for the construction of a variety of operator compositions and the development of new schemas of computation as well as network and computer architectures using operations with operators. Besides, operator representation allows for the efficient application of the axiomatic technique for the investigation of computation.

Keywords: computation; information; computing device; operator; network; computer architecture; composition

1. Introduction

The concept of an operator in mathematics stands for mapping or transformation from one space (set of elements) to another space. Linear maps are the basic operators on vector spaces. Linear operators (such as differentiation and integration) act on vector spaces of functions. The term operator is also used to denote a mathematical operation (like square root). In computer programming, a set of operators such as arithmetic-, comparison-, and logical-operators are supported by programming languages. The first textbook on operator theory [1] by Stefan Banach was published in 1932.

The concept of operator spans many disciplines as an important tool of theoretical studies and practical applications [2]. For instance, operators have become one of the most important tools in theoretical physics, especially quantum mechanics (see, e.g., [3,4]). In quantum mechanics, operators are represented as matrices, column vectors, and differential equations in an equivalent way. In mathematics, there is operator theory, which studies operators in Hilbert or Banach spaces (see, e.g., [5–7]). Operators are also becoming an important tool in information theory (see [8–11]). Many programming languages use operators. For instance, the programming language Python divides its operators into seven groups: arithmetic operators, assignment operators, comparison operators, logical operators, identity operators, membership operators, and bitwise operators [12]. The programming language Java divides its operators into five groups: arithmetic operators, assignment operators, comparison operators, logical operators, and bitwise operators [13].

Computing devices are information transformers and generators. That is why in this work, we develop operator models of computing devices and study their properties based on the unified operator theory [1].

2. Operator Modeling

In the context of this theory, we have the following definitions.

Definition O1. *An operator is an object (system) that operates, i.e., performs operations on some objects, systems, or processes, which are called operands of this operator.*

This brings us to the following definition.

Definition O2. *An operand is an object, system, or process operated by an operator.*

These definitions show that being an operator or an operand is a role and a characteristic of a system. One and the same system/object can be an operator in some situations and an operand in other situations. In a similar way, a system/object can be an operator with respect to some systems and not an operator with respect to other systems. All operators are systems, but not all systems are operators since subsequent to their formation, some systems can exist in substantial isolation from their environment to all intents and purposes.

Definitions O1 and O2 express the fundamental dyadic relation between operators and their operands, which is actualized in the form of the operator triad:

$$\text{Operator} \xrightarrow{\text{Operation/function}} \text{Operand}$$

The operator triad is a special case of the basic fundamental triad [14]. In the symbolic representation, the operator triad has the form

$$(\mathrm{Op}, \mathit{on}, \mathrm{Od})$$

where Op is an operator, *on* is an operation, and Od is an operand.

To construct a general mathematical operator theory in some domain, for example, in the realm of computations, it is necessary to organize the multiplicity of relevant operands in the form of an operating space, i.e., the space that is transformed by an operator.

Different types of operators function in distinct operating spaces. For instance, operators of quantum mechanics use Hilbert spaces as their operating space. Information operators function in information spaces. As there are diverse types of information, representation of them by operators demands different types of information spaces. Based on the general theory of information, we can differentiate syntactic, semantic, pragmatic, algorithmic, cognitive, emotional, and effective information spaces.

Consequently, modeling computing devices by information operators, we can treat them as tools for transformation and generation of distinct kinds of information. Here, we consider syntactic, semantic, and pragmatic information and corresponding information spaces.

In the case of digital computing devices, both physical and abstract, syntactic information spaces are systems of formal, artificial, or natural languages. For instance, taking such an abstract computing device as a Turing machine, we come to the syntactic information space, which consists of all formal languages with the alphabet of this Turing machine. In the case of finite automata, it is possible to utilize syntactic information spaces of three types. A syntactic information space of the first type consists of all symbols from the alphabet of the finite automaton. A syntactic information space of the second type consists of all symbols denoting states of the finite automaton. A syntactic information space of the third type consists of all formal languages with the alphabet of the finite automaton.

Digital computing devices process information in symbolic form, transforming words of some languages. Analog/physical computational processes are based on mapping between physical spaces. Consequently, both physical and abstract, syntactic information spaces are systems of formal, artificial, or natural languages. This implies that treating these devices as operators, we encounter operands of five types: (physical) data, separate symbols, words, texts, and languages and families of languages.

For instance, taking such an abstract computing device as a Turing machine, we come to the syntactic operating space, which consists of all formal languages with the alphabet of this Turing machine. At the same time, it is possible to take the space of all strings in some alphabet as the syntactic operating space of a Turing machine.

In the case of finite automata, it is possible to utilize syntactic information spaces of four types. A syntactic information space of the first type consists of all symbols from the alphabet of the finite automaton. A syntactic information space of the second type consists of all symbols denoting states of the finite automaton. A syntactic information space of the third type consists of all words from the formal languages with the alphabet of the finite automaton. A syntactic information space of the fourth type consists of all formal languages with the alphabet of the finite automaton.

The majority of abstract automata (computing devices) work with linear (i.e., one-dimensional) languages. However, there are also abstract automata (computing devices) that work with more sophisticated structures. For instance, Kolmogorov algorithms work with arbitrary graphs [15], Turing machines work with two-dimensional tapes, two-dimensional cellular automata work with two-dimensional structures, while structural machines work with arbitrary structures [16].

As the result, operators representing different abstract automata (computing devices) have different syntactic operating spaces.

Operators that represent Kolmogorov algorithms are Kolmogorov computation operators, the syntactic operating space of which is a collection of formal graph languages, i.e., languages words of which are graphs, while the definability domain consists of enumerable (recursively computable) graph languages [15].

Operators that represent two-dimensional cellular automata [17] are two-dimensional cellular computation operators, the syntactic operating space of which is the collection of two-dimensional array languages, i.e., whose language words are two-dimensional arrays, while the definability domain consists of all enumerable (recursively computable) two-dimensional array languages.

Operators that represent structural machines are structural computation operators, the syntactic operating space of which is the collection of structural languages, i.e., whose language words are structures, while the definability domain consists of all enumerable (recursively computable) structural languages.

Operators that represent Turing machines with one-dimensional tapes are one-dimensional Turing computation operators, the syntactic operating space of which is the collection of formal languages, while the definability domain consists of all recursively enumerable (recursively computable) languages.

Operators that represent Turing machines with two-dimensional tapes are two-dimensional Turing computation operators, the syntactic operating space of which is the collection of two-dimensional formal languages, while the definability domain consists of all recursively enumerable (recursively computable) two-dimensional languages.

Semantic information spaces can also have a different nature. For instance, in the semantic information theory of Bar–Hillel and Carnap, a semantic information space consists of possible worlds [18]. Often a semantic information space is a conceptual space, as in reference [19]. In the theory of epistemic information, a semantic information space is a conceptual space [9,10]. In the semantic information theory of Shreider, a semantic information space is a thesaurus as a system of texts and semantic relations between these texts [20]. Utilization of semantic information spaces in modeling of computing devices by information operators allows for the studying of semantic aspects of computation, computing systems, and networks.

Natural computation is physical computation performed through the dynamics of physical bodies [21], and involves information spaces of physical states (morphologies) of the systems undergoing spatiotemporal transformations [22–25]. In robotics, specific morphological computation that offloads computational tasks of control to the natural physical behavior and morphology of the robot body itself is presented, for example, by the authors of [26,27]. The conceptual model connecting subsymbolic (physical) and symbolic (formal language-based) levels of computation is given by Ehresmann [28].

3. Conclusions

To conclude, it is necessary to remark that operator modeling of computing devices—both abstract and physical—allows for the application of various constructions and results from the mathematical operator theory to automata, algorithms, and computations, presenting the following possibilities.

First possibility: Operator representation of computing devices allows for the formulating and solving of many computing device and network problems in a more general context in operating spaces of operators.

Second possibility: Operator representation of computing devices allows for the construction of a variety of operator compositions (operations) and the development of new schemas of computation as well as new network and computer architectures using operations with (composition of) operators.

Third possibility: Operator representation of computing devices allows for the efficient application of the axiomatic technique for the investigation of algorithms and computations.

Operator modeling of computing devices also presents various problems for further research. For instance, an interesting problem is to build a general mathematical description of operator spaces and to study operators in these spaces. One more interesting problem is the construction and exploration of operator algebras with nonstandard operations. There are different models of quantum computing involving dissimilar quantum theories [29]. Thus, an important task is to develop a unified operator theory of quantum computing.

Funding: One of the authors (GDC) has her research funded by Swedish Research Council grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Banach, S. *Théorie des Opérations Linéaires;* Monografie Matematyczne: Warszawa, Poland, 1932; Volume 1.
2. Burgin, M.; Brenner, J. Operators in Nature, Science, Technology, and Society: Mathematical, Logical, and Philosophical Issues. *Philosophies* **2017**, *2*, 21, doi:10.3390/philosophies2030021.
3. Dirac, P. *The Principles of Quantum Mechanics*; Oxford University Press: Oxford, UK, 1930.
4. Exner, P.; Havlíček, M. *Hilbert Space Operators in Quantum Physics*; Springer: New York, NY, USA, 2008.
5. Brown, A.; Pearcy, C. *Introduction to Operator Theory I: Elements of Functional Analysis*; Springer Verlag: New York, NY, USA; Heidelberg/Berlin, Germany, 1977.
6. Ball, J.A.; Bolotnikov, V.; Helton, J.W.; Rodman, L. (Eds.) *Topics in Operator Theory (Operator Theory: Advances and Applications)*; Birkhäuser Verlag: Basel, Switzerland, 2010.
7. Burgin, M. *Semitopological Vector Spaces: Hypernorms, Hyperseminorms and Operators*; Apple Academic Press: Toronto, NA, Canada, 2017.
8. Harris, Z. *A Theory of Language and Information: A Mathematical Approach*; Oxford University Press: Oxford, UK, 1991.
9. Burgin, M. Epistemic Information in Stratified M-Spaces. *Information* **2011**, *2*, 697–726.
10. Burgin, M. Weighted E-Spaces and Epistemic Information Operators. *Information* **2014**, *5*, 357–388.
11. Brenner, J.; Burgin, M. Information as a Natural and Social Operator. *Inf. Theor. Appl.* **2011**, *18*, 33–49.
12. Downey, A.B. *How to Think Like a Computer Scientist: Learning with Python*; Green Tea Press: Needham, MA, USA, 2002.
13. Eckel, B. *Thinking in Java*; Prentice Hall: Upper Saddle River, NJ, USA, 2006.
14. Burgin, M. *Theory of Named Sets*; Nova Science Publishers: New York, NY, USA, 2011.
15. Kolmogorov, A.N. On the Concept of Algorithm. *Russian Math. Surv.* **1953**, *8*, 175–176.
16. Burgin, M.; Adamatzky, A. Structural machines and slime mold computation. *Int. J. Gener. Syst.* **2017**, *45*, 201–224.
17. Codd, E.F. *Cellular Automata*; Academic: New York, NY, USA, 1968.
18. Bar-Hillel, Y.; Carnap, R. Semantic Information. *Brit. J. Phil. Sci.* **1958**, *4*, 147–157.
19. Gärdenfors, P. *Conceptual Spaces: The Geometry of Thought*; MIT Press: Cambridge, MA, USA, 2000.
20. Shreider, Y.A. On Semantic Aspects of Information Theory. *Inf. Cybern. (Moscow, Radio)* **1967**, *10*, 15–47. (in Russian).
21. Rozenberg, G.; Kari, L. The many facets of natural computing. *Commun. ACM* **2008**, *51*, 72–83.

22. Dodig-Crnkovic, G. Knowledge Generation as Natural Computation. *J. Syst. Cybern. Inf.* **2008**, *6*, 12–16.
23. Dodig-Crnkovic, G. Dynamics of Information as Natural Computation. *Information* **2011**, *2*, 460–477.
24. Dodig-Crnkovic, G. Nature as a Network of Morphological Infocomputational Processes for Cognitive Agents. *Eur. Phys. J.* **2017**, *226*, 181–195, doi:10.1140/epjst/e2016-60362-9.
25. Dodig-Crnkovic, G. Cognition as Embodied Morphological Computation. *Phil. Theory Artif. Intell.* **2017**, 19–23, doi:10.1007/978-3-319-96448-5.
26. Paul, C. Morphology and Computation. In Proceedings of the International Conference on the Simulation of Adaptive Behavior Los Angeles, CA, USA, 16–18 September 2004; pp. 33–38.
27. Pfeifer, R.; Iida, F. Morphological computation: Connecting body, brain and environment. *Jpn. Sci. Mon.* **2005**, *58*, 48–54.
28. Ehresmann, A.C. MENS, an Info-Computational Model for (Neuro-)cognitive Systems Capable of Creativity. *Entropy* **2012**, *14*, 1703–1716.
29. Freedman, M.H.; Kitaev, A.; Larsen, M.J.; Wang, Z. Topological Quantum Computation. *Bull. Am. Math. Soc.* **2002**, *40*, 31–38.