EasyChair Preprint

№ 6559

From Symbolic Computation to Super-Symbolic Computation

Mark Burgin and Rao Mikkilineni

September 8, 2021

From Symbolic Computation to Super-symbolic Computation

Mark Burgin[1] and Rao Mikkilineni[2]

[1] UCLA, Los Angeles, California, USA
[2] Golden Gate University, San Francisco, USA

Abstract:

Current 'state-of-the-art' information technologies (IT) utilize two forms of computation namely, symbolic computation and sub-symbolic computation. Symbolic computation is performed using algorithms on data structures, which contain knowledge about a particular state of the system in the form of a sequence of symbols. Sub-symbolic computation is performed by neural networks.

Biological systems also use both symbolic and sub-symbolic computations in the form of genes and neural networks. However, biological systems have evolved one step further by incorporating super-symbolic computation that performs computations on the combined knowledge from both symbolic and sub-symbolic computations to derive higher order autopoietic and cognitive behaviours.

The new type of computing automata called a structural machine provides means for modelling symbolic, sub-symbolic and super-symbolic computations performed on data and knowledge structures. In this work we argue that super-symbolic computation adds one more dimension to the general schema of computations. Synthesizing it with symbolic computation and sub-symbolic computation in one model, we come to symbiotic computation. Structural machines with flexible types of processors can accomplish symbiotic computation. Symbiotic computation combines advantages of sub-symbolic, symbolic and super-symbolic computations and advance the current state of the art IT with higher-order autopoietic and cognitive behaviours.

1. Introduction:

There are different types and forms of computations. According to the form of data processed, two pure forms of conventional computation are studied – symbolic computation and sub-symbolic computation (Burgin and Dodig-Crnkovic, 2015). These are pure types of computations while existing combined or amalgamated types and forms of computations are studied in Section 3.

Symbolic computation is performed with data having the form of explicit symbolic systems, such as systems of numbers, icons or letters, and the computing system operates with individual symbols. Here symbols are defined as linguistic objects and not in more general philosophical sense.

Sub-symbolic computation is performed as transformations of data described by non-linguistic objects such as specific signs, signals or geometric relationships. Note that sub-symbolic computations can be modeled by symbolic computations.

In both cases, the transformed entities are represented by the states of the computation system and by the states of its elements. Thus, to portray and study computation systems and their functioning, researchers use various mathematical models of computation, such as recursive functions or Turing machines, which work with separate symbols.

Sub-symbolic (intuitive) computation means that the machine (computation system) uses elementary operations with concealed semantics. Examples of such machines are neural networks and cellular automata. Sub-symbolic (intuitive) computation allows elimination of explicit algorithms/programs and using instead optimization processes, which improve functioning of the machine by upgrading implicit algorithms and programs of the machine.

Sub-symbolic (intuitive) computation is realized by the neural ensembles in the brain. Researchers of cognition conjecture that the object formation can function as the transition from a stream of massively parallel sub-symbolic micro-functional events to symbol-type, serial processing through sub-symbolic integration (Clark, 1989). Sub-symbolic (intuitive) computation is a model of functioning of the emotional and effective systems of the human brain (Burgin, 2010).

Symbolic (rational) computation means that the machine (computation system) uses elementary operations with elementary objects having individual semantics. Examples are Turing machines, inductive Turing machines, and vector machines (Burgin, 2005). Symbolic (rational) computation is a model of functioning of the left hemisphere of the human brain. The advantage of the symbolic (rational) computation is the explicit form of the algorithms and programs that control and direct the functioning of the machine.

2. New Pure Type of Computations:

At the same time, the general theory of structures and brain neurophysiology point to one more pure type of computation. Indeed, if there is sub-symbolic computation, then it must be *super-symbolic computation*, in which superstructures are transformed.

Researchers try to model functioning of the brain using artificial neural networks. It is possible to compare this to the situation when using only functioning of biological cells, biologists would try to explain the multifaceted functioning of the human organism with its higher functions.

Super-symbolic (transcendent) computation is a model of functioning of the right hemisphere of the brain. Processing of images by operation with holistic shapes is an example of super-symbolic computation. The advantage of the super-symbolic (transcendent) computation is its ability to operate big formal and informal systems of data and knowledge. Implementation of super-symbolic computation is the way to the solution of the problem of big data and information overflow.

Symbolic structures are composed from symbols in a simple way, that is, these structures have low structural complexity. Symbols, words, texts as linear composition of words, and sets are symbolic structures.

Symbolic superstructures are composed from symbols and symbolic structures. Intricate hypertexts, multicomponent images, and structures of higher order are symbolic superstructures.

3. Amalgamation of pure computational types:

Combination of pure types gives mixed types of information processing. The first step in this direction gives us *hybrid computation*, which comprises both symbolic and sub-symbolic computations being a two-fold type of computations (Burgin and Dodig-Crnkovic, 2015). Hybrid computation allows combining advantages of both symbolic and sub-symbolic computations.

Conventional models of computation perform either symbolic computation, e.g., finite automata, Turing machines, inductive Turing machines or Random Access Machines (RAM), or sub-symbolic computation, e.g., neural networks or cellular automata. New models, such as neural Turing machines (Graves, et al, 2014; Collier and Beel, 2018) or structural machines with symbolic and sub-symbolic processors, carry out hybrid computation.

A neural Turing machine is a recurrent neural network with a network controller connected to external memory resources. As a result, combines sub-symbolic computation of neural networks with symbolic computation of Turing machines.

Super-symbolic (intuitive) computation adds one more dimension to the general schema. Synthesizing it with symbolic (rational) computation and sub-symbolic (intuitive) computation in one model, we come to *symbiotic computation*. Structural machines with flexible types of processors can accomplish symbiotic computation. Symbiotic computation allows combining advantages of all three pure types of computation representing the entire type of computations.

In addition, there are two more twofold types of computations:

- *Fused computation* combines symbolic and super-symbolic computations.
- *Blended computation* merges sub-symbolic and super-symbolic computations.

As a result, we have three pure types, three twofold types and one entire type of computations.

Structural machines as a tool for amalgamated computations:

Structural machines allow even higher flexibility when they possess processors of different types (Burgin, 2020; Burgin and Mikkilineni, 2021). In particular, structural machines can have processors that work as neural networks, Turing machines or inductive Turing machines. Neural Turing machines are particular cases of structural machines.

A *structural machine M* works with structures of a given type and has three components:

- The *control device* C_M regulates the state of the machine *M* and can contain several components being distributed
- The *processor* P_M performs transformation of the processed structures and can contain several components (elementary processors) being distributed while their actions (operations) depend on the state of the machine *M* and the state of the processed structures
- The *functional space* Sp_M consists of three components:
    - The *input space* In_M , which contains the input structure.
    - The *output space* Out_M , which contains the output structure.
    - The *processing space* PS_M , in which the input structure(s) is transformed into the output structure(s).

We assume that all structures – the input structure, the output structure and the processed structures – have the same type.

Examples of heterogeneous distributed processors are processing devices in evolutionary automata such as evolutionary finite automata, evolutionary Turing machines or evolutionary inductive Turing machines (Burgin and Eberbach, 2009).

Cellular automata give examples of structural machines with (potentially) infinite distributed processors, in which unit processors are identical finite automata. One-dimensional cellular automata work with such structures as words. Two-dimensional cellular automata work with such structures as two-dimensional arrays.

It is possible to build structural machines that can work not only with discrete but also with continuous data because structures can be continuous and there are no restrictions on relations in processed structures. This possibility turns artificial neural networks, Shannon's differential analyzer (Shannon, 1941), a finite dimensional and general machine (Blum, et al, 1989) and Type 2 Turing machines (Weihrauch, 2000) into special cases of structural machines.

This shows that it is practical to discern discrete structural machines, which work with discrete structures, have discrete systems of states and operations, and continuous structural machines. In continuous structural machines one two or all three of the following components can be continuous, i.e., continuous processed structures, continuous system of states and/or continuous operations.

Note that structural machines can be:

- *single-level automata*, which process only data and include the majority of conventional automata, such as cellular automata, neural networks, Turing machines or inductive Turing machines
- *symmetric machines*, which process data and software (Schroeder, 2013; Burgin, 2020b)
- *balanced machines*, which process data and hardware (Dymond and Cook, 1980)
- *triadic machines*, which process data, software and hardware (Burgin, 2020a; Burgin, et al, 2020)

Note that depending on the type of its processors, a structural machine can be a single-level automaton, symmetric machines, balanced machines, or triadic machine.

Conclusion:

We have introduced and discussed a new pure type of computations called super-symbolic computation, three twofold types of computations called hybrid, blended and fused computations, and one entire type of computations called symbiotic computation.

Symbiotic computation combines advantages of sub-symbolic, symbolic and super-symbolic computations aimed at the advancement of the current state of the art IT with higher-order autopoietic and cognitive behaviors as well as at modeling of information processing in the mind.

Structural and functional analysis of these forms of computation is the further goal of this research.

References:

[1] Blum, L., Cucker, F., Shub, M., and Smale, S. *Complexity of Real Computation*, Springer, New York, 1998

[2] Burgin, M. *Super-recursive Algorithms*, Springer, New York/Heidelberg/Berlin, 2005

[3] Burgin, M. *Theory of Information*: *Fundamentality*, *Diversity and Unification*, World Scientific, New York/London/Singapore, 2010

[4] Burgin, M. *Information Processing by Structural Machines*, in Theoretical Information Studies: Information in the World, World Scientific, New York/London/Singapore, 2020, pp. 323–371

[5] Burgin, M. *Tri*adic Automata and Machines as Information Transformers, *Information*, v. 11, No. 2, 2020a, 102; doi:10.3390/info11020102

[6] Burgin, M. Information Processing by Symmetric Inductive Turing Machines, *Proceedings*, 2020b, v. 47, No.1, 28, 2 p. ; doi:10.3390/proceedings47010028

[7] Burgin, M. and Dodig-Crnkovic, G. A Taxonomy of Computation and Information Architecture, Proceedings of the 2015 *European Conference on Software Architecture Workshops*, Dubrovnik/Cavtat, Croatia, September 7-11, 2015, ACM, pp. 7:1-7:8

[8] Burgin, M. and Eberbach, E. *On Foundations of Evolutionary Computation*: *An Evolutionary Automata Approach*, in Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies (Hongwei Mo, Ed.), Section II: Natural Computing, Section II.1: Evolutionary Computing, Chapter XVI, Medical Information Science Reference/IGI Global, Hershey, Pennsylvania, 2009, pp. 342-260

[9] Burgin, M. and Mikkilineni, R. *From Data Processing to Knowledge Processing*: *Working with Operational Schemas by Autopoietic Machines*, Big Data Cogn. Comput. 2021, v. 5, 13 (https://doi.org/10.3390/bdcc5010013)

[10] Burgin, M., Mikkilineni, R. and Phalke, V. Autopoietic Computing Systems and Triadic Automata: The Theory and Practice, *Advances in Computer and Communications*, v. 1, No. 1, 2020, pp. 16-35 (also: EasyChair Preprint No. 4497, November 2, 2020, 15 p. (electronic edition: https://easychair.org/publications/preprint/Knkm))

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

[12] Collier, M. and Beel, J. Implementing Neural Turing Machines, *Artificial Neural Networks and Machine Learning* – ICANN 2018, pp. 94–104, 2018

[13] Dymond, P. W. and Cook, S.A. Hardware complexity and parallel computation, in 21st *Annual Symposium on Found. Comput. Sci*., Syracuse, New York, October 1980, pp. 360-372.

[14] Graves, A., Wayne, G. and Danihelka, I. Neural Turing Machines, arXiv:1410.5401, 2014

[15] Shannon, C. Mathematical Theory of the Differential Analyzer, *J. Math. Physics*, MIT, v. 20, 1941, pp. 337-354

[16] Schroeder, M.J. Dualism of Selective and Structural Manifestations of Information, in *Modelling of Information Dynamics*, Computing Nature, SAPERE 7, Springer, Berlin, Germany, 2013, pp. 125-137

[17] Weihrauch, K. *Introduction to Computable Analysis*, Springer, Berlin, 2000