Advances in Computer and Communication, 2020, 1(1), 16-35
http://www.hillpublisher.com/journals/acc/
ISSN Online: ****-****
ISSN Print: ****-****

Autopoietic Computing Systems and Triadic Automata: The Theory and Practice

Mark Burgin[1,*], Rao Mikkilineni[2], Vidya Phalke[3]

[1]Department of Mathematics, University of California, Los Angeles, USA.
[2]Ageno School of Business, Golden Gate University, San Francisco, USA.
[3]Chief Innovation Officer, MetricStream Palo Alto, USA.

How to cite this paper: Mark Burgin, Rao Mikkilineni, Vidya Phalke. (2020) Autopoietic Computing Systems and Triadic Automata: The Theory and Practice. *Advances in Computer and Communication*, 1(1), 16-35.
DOI: 10.26855/acc.2020.12.003

Received: October 20, 2020
Accepted: November 25, 2020
Published: December 8, 2020

[*]**Corresponding author:** Mark Burgin, Department of Mathematics, University of California, Los Angeles, USA.
Email: mburgin@math.ucla.edu

Abstract

Physical world contains many complex sentient structures. They have evolved to learn how to organize themselves and optimally use the resources available to them while interacting with their environment. These complex adaptive systems (CAS) sustain their continued existence in the face of external forces causing large fluctuations. The study of CAS functions, structures and their dynamics under the influence of fluctuations has thrown light into self-organizing patterns that are common among these disparate systems. Common theme among these structures is that a system encodes and processes information to organize and manage its components interacting with each other and their environment. The self-organizing patterns also sense and counteract fluctuations to maintain their stability. They become autopoietic and maintain homeostasis. Digital Computing structures composed of distributed and communicating software and hardware components also fall into the category of a complex system where fluctuations in the demand for, or the availability of, resources required to execute the computations disturb their stability and performance. The fluctuations impact the resiliency and efficiency of the structure as the scale of components increase. This paper describes the theory and practice of applying the self-organizing and self-managing patterns to distributed digital computing structures and making them autopoietic machines.

Keywords

Autopoiesis, Sentient Systems, Complex Adaptive Systems, Triadic Machines, Digital Genes, Digital Neurons, Distributed Computing, Church-Turing Thesis, Structural Machines, Knowledge Structures, Theory of Oracles

1. Introduction

All living beings exhibit sentience along with some form of intelligence and resilience. Sentience comes from the Latin sentient-, “feeling”, and it describes things that are alive, able to feel and perceive, and show awareness or responsiveness. The degree of intelligence (the ability to acquire and apply knowledge and skills) and resilience (the capacity to recover quickly from non-deterministic difficulties without requiring a reboot) depend on the cognitive apparatuses the organism has developed. The cognitive apparatuses are built using information processing structures that exploit physical, chemical and biological processes in the framework of matter and energy. These systems transform their physical and kinetic states to establish a dynamic equilibrium between themselves and their environment using the principle of entropy minimization.

According to Maturna and Varela [1], "A cognitive system is a system whose organization defines a domain of interaction in which it can act with relevance to the maintenance of itself, and the process of cognition is the actual (inductive) acting or behaving in the domain. *Living systems are cognitive systems, and living as a process is a process of cognition*. This statement is valid for all organisms, with or without a nervous system".

Cognition is a process of knowledge acquisition, understanding and organization. There are different forms of cognitive processes, which include perception, apprehension, learning, exploration, and investigation. One of the basic goals of cognition is adaptation to a certain environment. Thus, it is possible to treat cognition as one of the indispensable components of living. At the same time, artificial system can also have cognitive abilities. In essence, cognition is a principal part of intelligence.

It is possible to discern inner, outer and relational cognition. In inner cognition, the system acquires knowledge of itself while in the outer cognition it obtains knowledge about environment. Relational cognition is aimed at relations between the system and its environment.

Cognition emerges as a consequence of the continuous interaction between the components of the system and its environment. The continuous interaction triggers bilateral perturbations, which involve problems—therefore the system uses its functional differentiation procedures to come up with solutions to these problems. These solutions can be elaborated, found in the environment or in the system's memory. Difference between these situations determines the level of intellectual activity of the system.

Gradually, the system becomes "adapted" to its environment—that is it can confront the perturbations so as to survive. The resulting complexity of living systems is influenced by the cognition produced along the history of bilateral perturbations within the system/environment schema.

Cognition enables complex physical structures to evolve, adapt and become autopoietic. The term autopoiesis [1] refers to a system capable of reproducing and maintaining itself. "An autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components which: (i) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (ii) constitute it (the machine) as a concrete unity in the space where they (the components) exist by specifying the topological domain of its realization as such a network".

Evolution of self-maintenance patterns in complex adaptive systems (CAS) has been studied extensively and lessons learned have been successfully applied to understand and improve diverse domains [2-5] such as economics of stock market trading, modeling organizational change, supply-chain network dynamics, and examining global health governance. A CAS consists of a network of individual entities interacting with each other and its environment. Each entity exhibits a specific behavior and may be composed of subnetworks of entities providing a composed behavior. It takes energy to process information, sustain its structure and exhibit the intended behavior. Various systems adapt different strategies to use matter and energy to sustain order in the face of fluctuations caused by internal or external forces. The second law of thermodynamics comes into play because of matter and energy involvement which states that "there is no natural process the only result of which is to cool a heat reservoir and do external work". In more understandable terms, this law observes the fact that the useable energy in the universe is becoming less and less. Ultimately there would be no available energy left. Stemming from this fact we find that the most probable state for any natural system is one of disorder. All-natural systems degenerate when left to themselves.

An adaptive system refuses to be "left to itself" and develops self-organizing patterns to reconfigure the structure in order to compensate for the deviations of behavior due to fluctuations. Thus functions, structures, fluctuations, sensory perception, awareness and reconfiguration processes play key roles in the evolution of CAS.

Our paper has the following structure. In Section II, we analyze CAS and show how the theory of many-body interactions helps us to develop and understand its dynamics and state evolution. In Section III, we identify self-organizing and self-managing patterns and describe how they assist in sustaining order in the face of fluctuations. In Section IV, we discuss a new theory of triadic machines, which allows us to implement these self-organizing and self-managing patterns in digital computing systems. In Section V, we describe triadic structural machines as a mathematical model of self-organizing and self-managing information processing systems. In Section VI, we demonstrate that the current information processing structures implemented using John von Neumann's stored program architecture could be modeled as a complex adaptive system and that similar self-organizing and self-management patterns can be infused to sustain their behaviors in the face of fluctuations even as the system grows in scale and is geographically distributed. This observation is very timely to address the emerging problems with information processing networks, which are growing in scale connecting people, things and business process automation computing structures deployed in public, private, and hybrid clouds as well as in datacenters. In Section VII, we describe

application of triadic automata to the implementation of an autopoietic edge cloud where a distributed application workload is designed to manage itself in the face of fluctuations in the availability of or the demand for computing resources sustaining the computations. Section VIII concludes the paper with lessons learned identifying further research aimed at the improvement of the resiliency and efficiency at scale of how we deploy distributed computing structures in the future.

2. CAS and the Dynamics of Many-Body Interactions

"The emerging tapestry of complex systems research is being formed by localized individual efforts that are becoming subsumed as part of a greater pattern that holds a beauty and coherence that belies the lack of an omniscient designer. As in Navajo weaving, efforts on one area of this tapestry are beginning to meld into another, leaving only faint 'lazy lines' to mark the event". This poetic description from [6] summarizes the state of the art of our understanding of CAS bringing together many diverse disciplines. We have come a long way since the event in 1984, at Santa Fe Institute, which facilitated a new insight into the theory of complex adaptive systems and its application to multiple domains [7-10]. Instead of reviewing the current state of the art, in this paper, we take a new look at the CAS and its evolution using the new mathematical tools introduced by Mark Burgin [11] in the form of named sets, knowledge structures, cognizing oracle agents and structural machines.

We study CAS as a system composed of a named set of components interacting with each other and their environment where the local functions, global structure and fluctuations in their interactions play key roles in their evolutionary dynamics. Each component as a named component may also have its own structure that defines its internal and external interactions and evolution. This model of a complex system is a network of networks with the nodes defining the local functional behavior and links representing the interactions which, in turn determine the global dynamics of the system. Both locality of functions, their behaviors and the nature of interactions among themselves and their environment play a role on the overall structural behavior.

In this paper, we focus on a particular kind of structure which consists of a system of distributed components where each component itself may be composed of various concurrent asynchronous processes interacting with each other and their environment. The dynamics of such a system is non-deterministic based on the nature and the strength of fluctuations in the interactions. We focus on this type of structure because of its relevance to current day digital computing-based information processing structures and their efficiency and resiliency at scale in the face of fluctuations impacting the structure. When the fluctuations are small, the system usually exhibits near-equilibrium behavior and as the fluctuations increase in magnitude, the system may undergo chaotic behavior depending on the magnitude and the nature of fluctuations. Whether the trajectory of the system evolution is deterministic or non-deterministic, depends both on the nature of interactions and the strength of fluctuations. The nature of fluctuations determines the degree of non-linearity in the trajectory. Self-organizing and self-managing patterns are designed (often through evolutionary processes) to sense the nature of fluctuations and predict the impact on the system evolution and take action to counteract and maintain equilibrium. Biological systems manage homeostasis[1] through self-managing processes which are also implemented as physical structures.

Talking about the description of the physical world, Prigogine [12] points out the evolution of our theories from groups to semigroups, and from trajectories to processes. To him, it was evident from start that the physical structures were evolving out of fluctuations. "They appeared in fact as giant fluctuations, stabilized through matter and energy exchanges with the outer world. Since the formulation of the minimum entropy production theorem, the study of non-equilibrium fluctuation had attracted all his attention".

The structures are formed through the physical and chemical processes available in nature using matter, energy and their transformation rules. Atoms become molecules and molecules become compounds. Function, structure and fluctuations determine their macroscopic properties. For example, as the kinetic energy increases (because of heat from external source for example), the structure of a set of water molecules is rearranged going form solid form to liquid form or from liquid form to a gaseous form through physical processes. Same holds true for chemical structures when different physical structures interact with each other and form a composed structure using matter and energy transformations.

The rules that determine their transformations are well understood. Their evolution depends on the configuration of

[1] Homeostasis the tendency toward a relatively stable equilibrium between interdependent elements, especially as maintained by physiological processes. Example is how body maintains the temperature within a range when external temperature fluctuates.

the structure, strength of the interactions and the nature of fluctuations. Such systems can be represented by state vectors in phase space and their dynamics is determined by well-defined mathematical relationships that deal with matter, energy and their transformation rules defined by the physical processes. Mathematical representations of these structures stem from the rotational and translation invariance properties and the result of the complex space-time manifold.

In the physical world, as Prigogine pointed out, the fundamental triad of energy, structure and matter defines the system dynamics [12, 13]. Mark Burgin explains the relationship between matter, energy, knowledge and information as follows [14]. While knowledge and data are objects of the same type with knowledge being more advanced than data, information has a different type. It is possible to transform knowledge or data into information as we can transform matter into energy. Thus knowledge, structure and information form another triad which represents physical structures and their dynamics. Figure 1 shows the relationships between matter, energy, structure, information and knowledge.

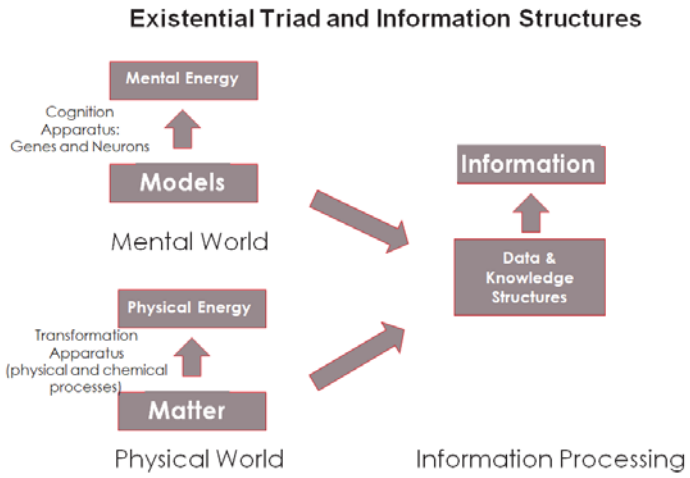

Figure 1. The relationship among matter, energy, information, knowledge and data.

The representation of physical structures is based on observation, a developed sense of awareness, classification, modeling, memory and reasoning which are characterized as cognitive abilities. A cognitive apparatus allows creating mental models and creating cognitive structures that represent observed physical structures and infer results about their evolution. Cognitive structures are possible only with cognitive apparatuses and biological systems have figured out how to create, use and replicate cognitive apparatuses. The genes in biological systems act as cognitive apparatuses, allow encoding information, configure physical structures and evolve them to execute cognitive processes using physical and chemical processes. The genome defines the blueprint to configure, monitor and manage a biological CAS with accumulated knowledge and its representation.

Complex adaptive structures therefore, are characterized by cognitive apparatuses that facilitate sentience (the ability to sense and feel), intelligence (the ability to process information) and resilience (using the information to rearrange the structure and facilitate its management using the cognitive apparatuses). The observations are represented in the form of named objects, their attributes (data) and the knowledge of the intra-object and inter-object relationships and behaviors when an interaction event perturbs any of the attributes. The mental world is represented by the cognitive apparatus that facilitates observation, modeling, memory, reasoning and action to rearrange the structures. The degree of cognition depends on the cognitive apparatuses developed and deployed in the system. The evolution of the cognitive apparatus is yet another physical structure, which allows encoding information and its processing mechanisms using physical and chemical processes, differentiates the sentient beings. As Charles Darwin explains “the difference in mind between man and the higher animals, great as it is, certainly is one of degree and not of kind” (cf., [15]).

Complex structures (with or without cognition) often exhibit emergent behavior, which arises from their interactions. In it, the collective behavior of various components in the structure is very different from the behavior that the components can produce separately. In essence, the whole is not just the sum of all the component behaviors. For example, the phase transition from water to ice or steam can be described as emergent. When a system undergoes a phase transition, its micro-components get rapidly reconfigured into a qualitatively different macro-structure. And

yet the components themselves are unchanged. Non-cognitive systems just are obeying the non-linear dynamics of the structure and its interaction with the environment influenced by the laws of nature. On the other hand, cognitive structures have developed various cognitive apparatuses to create sentient, resilient and intelligent behavior at scale. For example, in an ant colony, each ant is an autonomous unit that reacts depending only on its local environment and the genetically encoded rules for its variety of ant. Despite the lack of centralized decision making, ant colonies exhibit complex behavior and have even demonstrated the ability to solve geometric problems. For example, colonies routinely find the maximum distance from all colony entrances to dispose of dead bodies. Human genome provides the example of a complex set of cognitive apparatuses in the form of genes, neurons and cellular component structures with the highest degree of sentience, resilience and intelligence.

With the advent of digital computing machines, a whole new class of digital structures has evolved allowing information processing to extend our cognitive abilities of observation, modeling, memory, reasoning and action to rearrange the structures both in the physical and mental worlds. Figure 2 shows the relationship of digital structures to both physical and mental structures. This is made possible by John von Neumann's stored program implementation of the Turing machine (TM). It provides a physical implementation of a cognitive apparatus to represent and transform knowledge structures that are created by physical or mental worlds in the form of data structures representing the domain under consideration. Figure 2 represents the implementation of Turing Machines as a cognitive apparatus with locality and the ability to form information processing structures where information flows from one apparatus to another with a velocity defined by the medium. These implementations have allowed us to develop current state of the art of information processing structures using digital computing machines.

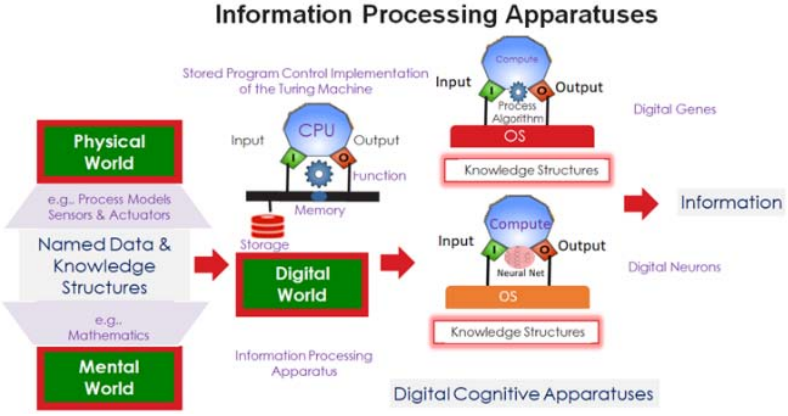

Figure 2. Structures as information processing machines.

In the physical world, the "gene" encodes the process of "life" in an executable form, and a neural network encodes various processes to interact with the environment in real time. Together, they provide the complex adaptive structures in the form of body, brain and the mind which exhibit sentience, resilience and intelligence.

The digital computing machine gives us the framework for encoding the models of physical and mental structures and their evolution in executable form where a digital computing structure provides information processing similar to the genes and neurons. Figure 2 depicts the digital gene and the neuron implemented using the digital computing machine. First, symbolic computing in the form of executable tasks allows us to process a list of formal, mathematical rules or a sequence of event driven actions such as modeling, simulation, business workflows, interaction with devices, etc. The digital computing machine, in essence, acts as a "cognitive apparatus" to implement cognitive functions described as easily described tasks. Second, algorithms are designed to mimic the neural networks in the brain and process information. The neural network model allows computers to understand the world in terms of a hierarchy of concepts to perform tasks that are easy to do "intuitively", but are hard to describe formally or a sequence of event driven actions such as recognizing spoken words or faces. Digital computing structures have allowed many paradigms of computation, including Mainframe, PC, Network, Internet, Distributed Computing, Grid Computing, Cloud Computing, Machine Learning and Deep Learning.

However, the limitation of current state of the art is pointed out by Cockshott et al. [16] in their book "Computation and its limits" with the concluding paragraph "The key property of general-purpose computer is that they are general purpose. We can use them to deterministically model any physical system, of which they are not themselves a part, to

an arbitrary degree of accuracy. Their logical limits arise when we try to get them to model the part of the world that includes themselves". A standard Turing machine is limited to single, sequential processes and is not amenable for expressing dynamic concurrent processes where changes in one process can influence changes in other processes while the computation is still in progress in those processes [17]. This is an essential requirement for describing cognitive processes. Concurrent and asynchronous task execution and regulation require a systemic view of the context, constraints, communication and control where the identities, autonomic behaviors and associations of individual components also must be part of the description.

The thesis of this paper is that sentience, resilience and intelligence are the results of information processing structures that various systems design and develop to manage their own state evolution in an optimal way by minimizing entropy in spite of a natural tendency for entropy to increase. They do this by creating physical structures that sense, model, monitor and evolve their states to establish equilibrium between their internal states and the environment with which they interact. Matter and energy together with physical, chemical and biological information processing structures that exploit their transformation rules enabled sentience, resilience and intelligence in the physical world.

We argue that we can model both physical and mental structures using structural machines [18-20], cognizing oracle[2] agents [21-23] and the knowledge structures [24, 25] along with the cognitive apparatuses (digital genes and neurons) that enable flow of information from one knowledge structure to another [26]. These digital structures implementing the structural machines allow us to design and develop sentient, resilient and intelligent systems with models that include both themselves and the physical structures they are made up of and interact with. In the next section we identify the key self-organizing and self-managing patterns extracted from the studies of CAS.

3. Self-Organizing and Self-Managing Patterns

Biological systems, in addition, have developed cognitive capabilities that allow them to:

- Integrate information from multiple sensory channels to marshal an effective response to fluctuating conditions;
- Make decisions under conditions of uncertainty;
- Share information within the components of the system and with outside, and
- Coordinating collective behavior to increase the chances of maintaining their desired state.

Cognition, in essence, allows the system to create a model of the physical structure of the self and its interactions with the environment in the form of mental structures and create additional physical processes that interact with physical structures to maintain its state or reconfigure itself to a new state that keeps it intact.

Biological systems have discovered a way to encode the processes and execute them in the form of genes, neurons, nervous system, the body and the brain etc., through evolutionary learning. The genome, which is the complete set of genes or the genetic material present in a cell or in an organism, defines the blueprint that includes instructions on how to organize resources to create the functional components, organize the structure and the rules to evolve the structure while interacting with environment using the encoded cognitive processes. Placed in the right environment, the cell containing the genome executes the processes that manage and maintain the self-organizing and self-managing structure adopting to fluctuations.

Shettleworth [28] studied cognitive apparatuses in animals and describes cognition in the following way. "Cognition refers to the mechanism by which animals acquire, process, store and act on information from the environment. These include perception, learning, memory, and decision making". She cites examples of crows using tools to crack nuts, bees communicating with dance, ants finding their way in the deserts and rats negotiating their way in mazes. "Cognitive is often reserved for declarative rather than procedural knowledge. Declarative knowledge is 'knowing that' whereas procedural knowledge is 'knowing how' or knowing what to do". Declarative knowledge implies more flexible behavior than the procedural knowledge, but in both cases behavior results from processing and storing information about the world. "A related distinction is that between first-order and higher-order processes, only the latter of which may be regarded as interestingly cognitive. First-order processes operate directly on perceptual input, as when a stimulus triggers a response or creates a trace in memory. Second-order processes operate on first-order processes, as in evaluating the strength of one's memory for an event".

These observations are very relevant to the digital information processing structures in attempting to infuse cognition into them. The digital genes and the digital neurons mentioned above generate the first order in intelligent processes on the Internet. In addition, the cognitive agent behavior described in the theory of oracles [24] is

[2] Theory of oracles built by Mark Burgin [23] is a far-reaching generalization of the concept proposed by Alan Turing [27].

second-order intelligent process, which operates on the first-order processes. The next two sections present the theory of triadic automata (TA) where three-level digital computing structure is describes that points a way to infuse cognitive behavior in digital information processing systems.

4. Triadic Automata and Machines as Mathematical Models of Autopoietic Machines

An autopoietic machine is an appealing inspiration but it is only an idea and to put it into practice demands transformation of this idea into a functioning physical machine. The process of transformation usually includes several stages.

First, it is necessary to develop a realistic structure of an autopoietic machine.

Second, it is necessary to specify components of the developed structure.

Third, it is necessary to construct a physical realization of the autopoietic machine.

To make this process constructive, we need to analyze these stages and what they involve.

In this context, the structure can be informal or formal. For construction, the structure must be formal and its formalization can be mathematical or engineering. The same is true for specifications. They can be informal or formal while formal specifications can be mathematical or engineering.

Some think that it is simpler to proceed from a general idea to the engineering descriptions of the future system. However, peoples experience shows that engineering projects are always better when they are based on sound mathematical theories and models. Indeed, theoretical grounding and mathematical modeling allows evading many mistakes in engineering development as well as obtaining a better, e.g., more efficient or optimal, system.

One of the most transparent examples for this is the history of the von Neumann architecture.

It tells us that the earliest computing machines had fixed programs. Changing the program of a fixed-program machine required rewiring, restructuring, or even redesigning the machine. It was—when possible at all—an arduous process. It started with producing paper notes and flowcharts, followed by detailed engineering designs, and then the painstaking process of physically rewiring and rebuilding the machine. In particular, it could take up to three weeks to set up and debug a program on one of the first computers ENIAC.

In 1945, while consulting for the Moore School of Electrical Engineering on the EDVAC project, von Neumann wrote an incomplete set of notes, titled the First Draft of a Report on the EDVAC. These notes were widely distributed laying foundations of a novel, at that time, computer architecture in which the data and the program are both stored in the computer's memory. It was called later von Neumann architecture, which became the de facto computer standard for a long time and sometimes is still used today.

The main idea of von Neumann architecture was copied from such a mathematical model of computation as a universal Turing machine. Although von Neumann did not mention this in his Report on the EDVAC, he was well acquainted with the work of Turing, who was a graduate student at Princeton when he published his famous paper describing what was later called Turing machine [29].

That is why here we build a mathematical model of an autopoietic machine using the pinnacle achievements and the most advanced structures of computer science. It is important to note that till now, nobody elaborated such a model although there are some predecessors of it. In particular, self-reproductivity was the principal feature of cellular automata introduced by von Neumann [30].

To build an adequate mathematical model of an autopoietic machine, we need to analyze the idea of such a machine and the tentative informal description of its functioning. In essence, an autopoietic machine is a technical system capable of regenerating, reproducing and maintaining itself by production, transformation and destruction of its components and the network of processes in these components.

When we have a technical (physical) information processing system, its inner structure comprises three components: hardware, software and infware [31].

The *hardware* of a system consists of all devices that belong to the system and connections between them.

The *software* of a system consists of various programs and algorithms that control its functioning.

The *infware* of a system consists of diverse information carriers, e.g., data or knowledge, with which this system works.

Thus, the inner structure of an information processing system consists of these components and relations between them [31].

In a general situation, the hardware of an information processing system also has three key components: *the input subsystem*, *output subsystem* and *processing subsystem*. If we want to build an adequate mathematical model of a

physical system with these components, we need to establish the corresponding components in the model, which is an abstract automaton or machine. However, in many theoretical models of computation, such as finite automata or conventional Turing machines, input and output devices are either not specified or represented by components of the common memory and/or of the processor. For instance, a conventional Turing machine performs input and output operations by writing the input to and reading the output from the working memory, which consists of one or several tapes. In contrast to this, any inductive Turing machine has special input and output registers for this purpose, e.g., tapes [31].

Neural networks also have these three key components: their input subsystem comprises all input neurons while the output subsystem consists of all output neurons. At the same time, it is possible to define its information processing subsystem either as all neurons of the network or only all its *hidden neurons* [32]. Some authors call input and output neurons by the name *visible neurons*.

Considering infware, we see that the majority of abstract automata (abstract computing devices) work with strings of symbols, and thus, their infware consists of strings in some alphabet.

Turing machines with two-dimensional tapes and two-dimensional cellular automata work with two-dimensional symbolic arrays. It means that their infware consists of two-dimensional symbolic arrays. Turing machines with n-dimensional tapes and n-dimensional cellular automata work with n-dimensional symbolic arrays. It means that their infware consists of n-dimensional symbolic arrays.

At the same time, such algorithms (automata) as Kolmogorov algorithms and storage modification machines work with arbitrary graphs [33]. In addition, there are also many concrete algorithms that work with graphs (cf., for example, [34, 35]). Consequently, their infware also consists of graphs.

Structural machines work with arbitrary structures [18-20]. Consequently, their infware consists of arbitrary structures.

Coming to software, we see that in many abstract automata, such as finite automata, pushdown automata, Minsky machines, timed automata, register machines, Kolmogorov algorithms, random access machines (RAM), and Turing machines, instructions and rules compose their software.

Neural networks are often contrasted to other kinds of abstract automata. However, it is possible to treat the system of weights, activation functions, threshold functions and output functions as the software of neural networks. It is possible to treat these systems as algorithms although their form is different from traditional algorithms, which are described as sets of instructions.

Existence of three components in the automaton or machine implies that to be autopoietic, a machine needs ability to perform operations with these components. The unique theoretical model of computation that can do this is a *triadic automaton* or *machine* [36]. Let us describe this model.

Triadic automata (machines) transform infware (for example, data), software (for example, instructions or programs) and hardware (for example, memory or processors). There are different types of triadic machines (triadic automata):

- a *hardware modification machine (automaton)* transforms only infware and hardware
- a *software modification or symmetric machine (automaton)* transforms only infware and software
- a *full triadic machine (automaton)* transforms infware, software and hardware
- a *transducer* transforms only infware and has input and output
- a *generator* transforms only infware and has only output
- an *acceptor* transforms only infware and only input
- a *hardware expansion machine (automaton)* only expands its hardware
- a *software expansion machine (automaton)* only expands its software
- a *symmetric expansion machine (automaton)* only expands its hardware and software
- a *hardware alteration machine (automaton)* only updates its hardware
- a *software alteration machine (automaton)* only updates its software
- a *symmetric alteration machine (automaton)* only updates its hardware and software

Besides, there are different ways to perform hardware/software modifications. With respect to the source of modification it is possible to consider three types of hardware/software modifications in an automaton (machine) M:

- *External modification* is performed by another system.
- *Internal modification* is performed by the automaton (machine) M.

– *Combined modification* is performed by both the automaton (machine) M and another system.

Taking a specific class of triadic automata (machines), we see that what modifications are possible and permissible in a given class depends on the structure of a triadic machines (triadic automata) from this class. In any case, mandatory components, which make these automata efficient for computation, include input and output systems working together with one or more processors. Often, input and output components of an automaton (machine) are specific registers in the memory of the machine (automaton) [31]. At the same time, input and output in neural networks are organized using specified neurons [32].

At the same time, adding memory and other components to automata allows increasing their flexibility, interoperability and efficiency. These changes are reflected in the structure of triadic machines (automata) of different types. Let us consider two of such types - state and instruction triadic automata (machines).

Definition 4.1. A *state triadic machine* or *triadic state automaton A with memory* has seven core hardware components:

- The *control device* C_A, which is a finite automaton and represents states of the machine (automaton) A
- The *data memory* W_A, which stores data and includes input and output registers
- The *software memory* V_A, which stores software of the machine (automaton) A
- The *data processor* P_M, which transforms (processes) information (data) from the memory W_M
- The *software processor* D_M, which transforms (processes) software of A stored in the memory V_M

The *metaprocessor* P_A, which transforms (e.g., builds or deletes connections in) the hardware H_A and/or changes the control device C_A

In the standard form, both memories consist of cells, which are connected by transition links. Processors have their programs of functioning, which constitute the software of the automaton.

In the same way as state triadic machines, instruction triadic machines constitute a special class of triadic machines. In a general case, it is possible that the functioning of an instruction triadic machine does depend on its state. However, we include the state system in the general description of instruction triadic machines because when the functioning of an instruction triadic machine does depend on its state, it is possible to treat this as a machine with only one state.

Definition 4.2. A *instruction triadic machine* or *instruction triadic automaton H with memory* has seven core hardware components:

- The *control device* C_H, which is a finite automaton and represents states of the machine (automaton) H
- The *data memory* W_H, which stores data
- The *instruction memory* V_H, which stores instructions
- The *data processor* P_M, which transforms (processes) information (data) from the memory W_M
- The *instruction processor* D_M, which transforms (processes) information (instructions) from the memory V_M
- The *memory processor* P_W, which transforms (builds or deletes connections and/or cells in) the memory W_M
- The *memory processor* P_V, which transforms (e.g., builds or deletes connections and/or cells in) the memory V_M

Memory processors are hardware transformers and it is also possible to include a control device processor in the structure of an instruction triadic machine. This additional processor changes the control device C_A.

Many kinds of algorithms and abstract automata, such as finite automata, pushdown automata, register machines, Kolmogorov algorithms, random access machines (RAM), and Turing machines, use systems of instructions, for example, in the form of transition rules, to control computational processes. These instructions determine computational processes, which are controlled by algorithms and are going in these automat and constitute the software of these automata and machines, which form specific classes of instruction machines.

There are different classes of instruction triadic machines (automata). Namely, when instruction triadic machines (automata):

– do not have processors that transform hardware, e.g., memory or processors, they are called *symmetric instruction machines*

– have only processor(s) that transform infware, e.g., data, they are called *pure instruction machines*

– have only processor(s) that transform software, i.e., systems of instructions, they are called *translation machines* or *translators*

– have only processor(s) that transform hardware, i.e., memory or processors, they are called *construction machines* or *constructors*

– do not have processors that transform infware, e.g., data, they are called *constructors* (*construction machines*)

with translators

– do not have processors that transform software, i.e., systems of instructions, they are called *generative instruction machines*

Machines from each class have their specific functions. For instance, construction machines (constructors) can be used to construct memory for other machines. This technique is employed in inductive Turing machines of the second and higher orders use inductive Turing machines of lower orders as their constructors [31, 37].

Besides, there are different methods to organize program formation with the help of computing/constructing agents. If the memory of the automaton has connections between any pair of cells, then the program can use these connections. Thus, it is possible to organize the inductive mode of computing by inductive computation (compilation) of the program for the main computation.

In the simplest approach called the *sequential strategy*, it is assumed that given some schema, for example, a description of the structure of the memory E of an inductive Turing machine M, an automaton A builds the program and places it in the memory E before the machine M starts its computation. When M is an inductive Turing machine of the first order, its constructor A is a Turing machine, which, for example, puts the names of the connections of the memory of M into instructions (rules) of M. When M is an inductive Turing machine of the second or higher order, its constructor A is also an inductive Turing machine, the order of which is less than the order of M and which modifies instructions (rules) of M. For instance, the program of inductive Turing machines of the second order is constructed by Turing machines of the first order

According to another methodology called the *concurrent strategy*, program formation by the automaton A and computations of the machine M go concurrently, while the machine M computes, the automaton A constructs the program in the memory E.

It is also possible to use the *mixed strategy* when some parts of the program E are assembled before the machine M starts its computation, while other parts are formed parallel to the computing process of the machine M.

These three strategies determine three kinds of the constructed program (software):

- In the *static program* (*static software*) of the machine M, everything is constructed before M starts working.
- In the *growing program* (*growing software*) of the machine M, parts are constructed while M is working but no parts are deleted.
- In the *dynamic program* (*growing software*) of the machine M, when it necessary, some parts are constructed and when it necessary, some parts are deleted while M is working.

It is possible to use similar strategies for hardware modification. This approach determines three types of the constructed hardware of a triadic automaton/machine:

- In the *static hardware* of the machine M, everything is constructed before M starts working.
- In the *growing hardware* of the machine M, parts are constructed while M is working but no parts are deleted.
- In the *dynamic hardware* of the machine M, when it necessary, some parts are constructed and some parts are deleted while M is working.

In the next section, we discuss problems of efficiency of information processing describing the most flexible and efficient model of computation, which is called structural machine.

5. Triadic Structural Machines as Advanced Forms of Triadic Automata

Information is represented by diverse structures. Only some of them are formalized as data structures and used in real-life computations. However, conventional theoretical models even do not include the majority of data structures operated by contemporary computers. Indeed, programming languages use a variety of data structures such as characters, integers, floating-point real number values, enumerated types (i.e., a small set of uniquely-named values), arrays, records (also called tuples or structs), unions, lists, streams, sets, multisets, stacks, queues, double-ended queues, trees, general graphs, etc. In addition, word processors, such as Word or TeX, work with various geometrical shapes, figures and pictures.

At the same time, data structures processed by abstract automata are more limited because they work only with separate symbols, strings of symbols, trees and arrays. To eliminate this limitation, the new model of computation called structural machine was introduced [18-20]. These machines provide means of processing arbitrary structures. Here we introduce new forms of these machines – triadic structural machines and autopoietic structural machines.

An *autopoietic structural machine* is a technical system capable of regenerating, reproducing and maintaining itself by production, transformation and destruction of its components and the network of processes in these components by working with a variety of flexible structures.

Triadic structural machines are mathematical models of autopoietic structural machines.

Here for simplicity, we consider only structural machines, which work with structures of the first order.

Definition 5.1 [38]**.** A *first-order structure* is a triad of the form

$\boldsymbol{A} = (A, r, \boldsymbol{R})$

Here:

- A is a set, which is called the *substance* of the structure $\boldsymbol{A}$ and consists of elements of the structure $\boldsymbol{A}$, which are called *structure elements* of the structure $\boldsymbol{A}$
- $\boldsymbol{R}$ is a set, which is called the *arrangement* of the structure $\boldsymbol{A}$ and consists of relations between elements from A in the structure $\boldsymbol{A}$, which have the first order and are called *structure relations* of the structure $\boldsymbol{A}$
- r is the *incidence relation*, which connects groups of elements from A with the names of relations from $\boldsymbol{R}$

For instance, if $\boldsymbol{R}$ is an n-ary relation from $\boldsymbol{R}$ and $a_1 , a_2 , a_3 , \dots , a_n$ are elements from A, then the expression $r(R; a_1 , a_2 , a_3 , \dots , a_n)$ means that the elements $a_1 , a_2 , a_3 , \dots , a_n$ belong to the relation $\boldsymbol{R}$ with the name R, i.e., the incidence relation r connects the elements $a_1 , a_2 , a_3 , \dots , a_n$ with R. The number n is called the arity of $\boldsymbol{R}$ and is denoted by $\alpha(R)$.

It is necessary to remark that all conventional concepts of structure include only structures of the first order (cf., for example, [39-41]. Only Bourbaki go to higher structures in their formal definition. However, they are confined to the set-theoretical context and use unnecessary conditions making their definition blurred and too restrictive [42, 43]. The comprehensive definition of structures of all orders is elaborated in the general theory of structures [38].

Describing structures, it is important to distinguish relations and their names because when a structural machine functions, relations, as a rule, are changing, while their names can remain the same indicating that this relation is dynamically the same but only some of its properties are changing. For instance, when a structure $\boldsymbol{A}$ on a set A has a ternary relation $\boldsymbol{R}$ with the name R as its component and in the process of computation, the machine connects three elements that were not related by $\boldsymbol{R}$ by a link including this triple into the relation $\boldsymbol{R}$. As a result, $\boldsymbol{R}$ becomes larger but stays dynamically the same preserving the same name R. This is similar to the situation when a human being is changing remaining, at the same time, the same person.

Lists, queues, arrays, words, texts, graphs, multigraphs, directed graphs, mathematical and chemical formulas, tapes of Turing machines and Kolmogorov complexes are particular cases of structures of the first order that have only unary and binary relations. Note that labels, names, types and properties are unary relations.

In the case when the arrangement $\boldsymbol{R}$ of relations consists of one binary and several unary relations, the first order structure is a labeled (named) directed graph. Transition diagrams of finite automata are examples of labeled directed graph. When the arrangement $\boldsymbol{R}$ contains only binary and unary relations, the first order structure is a labeled (named) directed multigraph [44]

Definition 5.2. The *type* $\mathrm{T}(\boldsymbol{A})$ of a first-order structure $\boldsymbol{A} = (A, r, \boldsymbol{R})$ is the set $\{(R, \alpha(R)); \boldsymbol{R} \in \boldsymbol{R}\}$ of pairs $(R, \alpha(R))$ where $\alpha(R)$ is the arity of, i.e., the number of variables in, the relation $\boldsymbol{R}$ with the name R.

We assume that two first-order structures $\boldsymbol{A} = (A, r, \boldsymbol{R})$ and $\boldsymbol{B} = (B, p, \boldsymbol{P})$ have the *same type* if there is a one-to-one mapping f: $\mathrm{T}(\boldsymbol{A}) \to \mathrm{T}(\boldsymbol{B})$ such that if $f(R, \alpha(R)) = (P, \alpha(P))$, then $\alpha(R) = \alpha(P)$.

For instance, all binary relations have the same type.

Definition 5.3. A structural machine M works with structures of a given type and has three components:

- The *unified control device* C_M regulates the state of the machine M
- The *unified processor* P_M performs transformation of the processed structures and its actions (operations) depend on the state of the machine M and the state of the processed structures
- The *unified functional space* Sp_M contains input, output and processed structures and 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).

In many cases, it is assumed that all structures – the input structure, the output structure and the processed structures – have the same type.

Definition 5.4. A structural machine M is *triadic* if it is a triadic automaton, i.e., it processes its infware, software and hardware.

Computation of a triadic structural machine M determines the *trajectory of computation*, which is a tree in general

case and a sequence when the computation is deterministic case and is performed by a single processor unit.

Definition 5.5. There are three forms of unified functional spaces - CSp_M , TSp_M and USp_M - in structural machines in general and triadic structural machines in particular:

- CSp_M is called a *categorical functional space* and is the set (category) of all structures that can be processed by the (triadic) structural machine M
- USp_M is called a *universal functional space* and is a structure for which all structures that can be processed by the (triadic) structural machine M are substructures of USp_M
- TSp_M is called a *transformation functional space* and is a structure for which all structures that can be processed by the (triadic) structural machine M are transformations of TSp_M

Definition 5.6. There are three basic types of unified control devices:

- A *central control device* controls all processors of the triadic structural machine
- A *cluster control device* controls a cluster of processors in the triadic structural machine
- An *individual control device* controls a single processor in the triadic structural machine

Definition 5.7. There are three basic types of unified processors:

- A *localized processor* is a single abstract device (*processor unit* or *unit processor*)
- A *distributed processor*, which is also called a *total processor of the first level*, consists of a system of *unit processors* or *processor units*
- A *clustered processor*, which is also called a *total processor of the second level*, consists of a system of *total processors of the first level*

Continuing this construction, we can define and build total processors of any positive level in a triadic structural machine.

It is possible to treat a localized processor in a triadic structural machine as a singular unit processor although it can be constructed as a robustly tied together several processor units.

Note that a triadic structural machine can have several distributed and/or clustered processors.

In turn, there are three basic types of distributed processors in a triadic structural machine:

- A *homogeneous distributed processor* consists of a system of identical unit processors, i.e., all these unit processors are copies of one processor
- An *almost homogeneous distributed processor* consists of a system in which almost all unit processors are identical
- A *heterogeneous distributed processor* consists of a system of different unit processors

As a result, we have three *structural types* of processors in a triadic structural machine.

There are also three basic *temporal classes* of distributed processors in a triadic structural machine:

- In a *synchronous distributed processor*, all unit processors perform each operation at the same time
- In an *almost synchronous distributed processor*, almost all unit processors perform each operation at the same time
- In an *asynchronous distributed processor*, unit processors function independently

A cellular automaton is a synchronous distributed processor, while a Petri net is an asynchronous distributed processor [45].

Organization of control in distributed processors determines three *control categories* of distributed processors in a triadic structural machine:

- A distributed processor with the *centralized control* has one control device, which controls all its unit processors
- A distributed processor with the *dispersed control* has several control devices, each of which controls a group of its unit processors
- A distributed processor with the *individualized control* has a control device for each of its unit processors

In a general case, a unit processor moves from one topos, e.g., a structure element of infware, software or hardware, to another performing operations in their neighborhoods. It is possible to assume that unit processors of a triadic structural machine perform only local operations. As a result, it is possible to consider a localized processor as a special type of a distributed processor with one unit processor. A model example of a localized processor is the head of a Turing machine with one head or a finite automaton.

A model example of a homogeneous distributed processor is the system of all heads in a Turing machine with several heads. Examples of heterogeneous distributed processors are processing devices in evolutionary automata

such as evolutionary finite automata, evolutionary Turing machines or evolutionary inductive Turing machines [46].

In a general case, not all heterogeneous distributed processors are the same and to discern their properties it is possible to use measures of homogeneity [47]. These measures can be also useful for exploring and utilizing structures processed by computers and their models such as structural machines.

As triadic structural machines process structures from all its three basic components, it is natural to specialize processors of these machines. This gives us three pure types of processors in a triadic structural machine:

- *Infware processors* or I-*processors* Pr_I work with infware
- *Software processors* or S-*processors* Pr_S work with software
- *Hardware processors* or H-*processors* Pr_H work with hardware

Definition 5.8. A triadic structural machine *M* is *stratified* if all its processors are purely specialized.

There are also four general types of processors—flexible processors and three types of mixed processors in a triadic structural machine:

- *Flexible processors* or F-*processors* work with infware, software and hardware
- IS-*processors* work with infware and software
- IH-*processors* work with infware and hardware
- SH-*processors* work with software and hardware

For a general purpose processor, its specialization is defined by the role it plays in the computational process. Pure-type processors can play only one role. Mixed processors can play two roles while a flexible processor can play all three roles.

In the process of computation, a metaprogram of a triadic structural machine determines the role of each processor according to its restrictions and possibilities. Metaprograms of triadic structural machines are second-level algorithms, which contain not only instructions for data processing but also instructions directing application of instructions for data processing [48, 49].

A further specialization determines the part of the system components, with which the processor works. With respect to this specialization, S-processors, H-processors and SH-processors can be:

- *module* H-*processors* (S-*processors*), which create/add, destroy/eliminate or transform/change modules, components, and elements of the system hardware (software).
- *connection* H-*processors* (S-*processors*), which create/add, destroy/eliminates open or close connections between modules, components, and elements of the system hardware (software).
- *module-module* SH-*processors*, which create/add, destroy/eliminate or transform/change modules, components, and elements of the system hardware and software.
- *module-connection* SH-*processors*, which create/add, destroy/eliminates open or close connections between modules, components, and elements of the system software and create/add, destroy/eliminates open or close connections between modules, components, and elements of the system hardware.
- *connection-module* SH-*processors*, which create/add, destroy/eliminates open or close connections between modules, components, and elements of the system hardware and create/add, destroy/eliminate or transform/change modules, components, and elements of the system hardware.
- *connection-connection* SH-*processors*, which create/add, destroy/eliminates open or close connections between modules, components, and elements of the system hardware and software.

Note that changing connections, it is possible to add or eliminate parts of the system hardware or software.

With respect to its orientation, there are three types of pure processors:

- an *external* S-*processor* (H-*processor*) transforms software (hardware) of another processor from its own triadic structural machine
- an *internal* S-*processor* (H-*processor*) transforms its own software (hardware)
- an *outer* S-*processor* (H-*processor*) transforms software (hardware) of another triadic structural machine

There are also nine types of mixed processors:

- an *external* SH-*processor* transforms software and hardware of another processor from its own triadic structural machine
- an *internal* SH-*processor* transforms its own software and hardware
- an *outer* SH-*processor* when it transforms software and hardware of another triadic structural machine
- an *ex-internal* SH-*processor* transforms software of another processor from its own triadic structural machine and

its own hardware

- an *in-external* SH-*processor* transforms its own software and hardware of another processor from its own triadic structural machine
- an *o-internal* SH-*processor* transforms software of another triadic structural machine and its own hardware
- an *o-external* SH-*processor* transforms software of another triadic structural machine and hardware of another processor from its own triadic structural machine
- an *ex-outer* SH-*processor* transforms software of another processor from its own triadic structural machine and hardware of another triadic structural machine
- an *in-outer* SH-*processor* transforms its own software and hardware of another triadic structural machine

The developed classification of control devices and processors of triadic structural machines is aimed at the analysis, exploration and utilization of properties of these machines for the development of future autopoietic supercomputers and autopoietic computer networks.

6. CAS, Digital Information Processing Structures and Current Information Technology (IT) Evolution

In summary, a complex adaptive system is a dynamic structure which can be represented as a network of networks connecting dynamic functional behaviors (where individual components interact with each other and their environment) using information processing nodes and links. The network of network implies that nodes contain nodes and links may contain links. The nature and strength of these interactions define the complexity. In addition, fluctuations in these interactions often result in the properties of emergence where the system behavior as a whole is not just equal to the sum of the behaviors of the individual nodes. Physical and chemical systems process information using physical and chemical processes obeying the laws of physics. On the other hand, biological systems have developed cognitive apparatuses in the form of genes and neurons to model the systemic behaviors, monitor them and manage them using their cognitive apparatuses with embedded, embodied, enactive and extended (4E) cognition. Genes enable encoding the information processing structures, their execution mechanisms and their day to day operation. Neurons in the nervous system or the brain provide the information processing neural networks that assist in building cognitive behaviors sensing, interacting and managing information processing through memory and reasoning apparatuses. Cognition, thus is a biological phenomenon. Biological systems have developed ways to leverage 4E cognition to create higher level cognitive behaviors as colonies, societies etc., thus endowing themselves with higher degree of resilience and efficiency at scale. It is important to note that emergence is non-deterministic with an element of surprise, while cognition is an instrument to control the dynamics to reach a new state of the structure by rearranging it to meet the goals with resiliency and efficiency.

Digital Computational Structures, Computing Models, Models of Computation, Sentient Behaviors and their resiliency and efficiency optimization:

With the advent of digital genes and neurons discussed earlier, information processing structures in the digital realm have in a sense "exploded" projecting their impact on the Internet and the current Cloud Computing technologies [50]. The novel technologies, which are described above, allow us to model both the physical structures and the mental structures, reason about them, and manage them by establishing sensors and actuators to interact with the physical world. Figure 3 depicts the relationships between the physical, mental and cognitive structures.

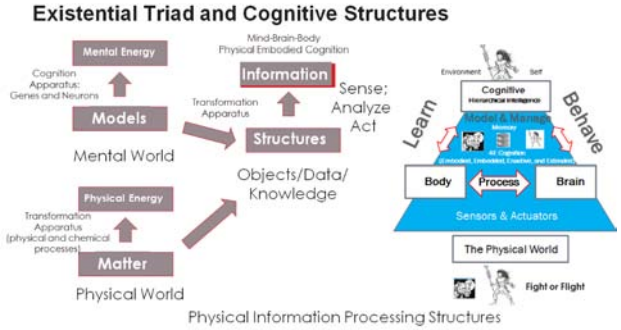

Figure 3. Physical and Mental Structures, their representation in the Digital Structures and Cognitive Modeling, Reasoning and self-Managing Patterns with Mind-Brain-Body Embodied Cognition.

In biological systems, cognition is embodied in physical structures and is encoded in its cell that enables its functions, structure and fluctuation management. As Waldrop Mitchell points out in his book on complexity [10], "The DNA was actually the foreman in charge of construction. In effect, DNA was a kind of molecular-scale computer that directed how the cell was to build itself and repair itself and interact with the outside world". Each cell can divide and differentiate itself into muscle cells, brain cells, liver cells, and all other kinds of cells that make up a new born. Each different type of cell corresponds to a different pattern of activated genes. In Figure 3, we represent the embodied cognition in the human genome that senses, models, reasons and manages both physical and mental structures (using the mind, brain, and body structures).

On the other hand, digital structures are made possible by the creativity of the human mind and its implementation of cognitive representations of the physical world using physical structures such as computers, networks and storage devices. Current information technologies are implemented as physical structures representing the models of various physical and mental structures using the digital genes and digital neurons. They are modeled and managed by the people. Figure 4 shows the digital structures and their implementation based on physical structures such as computers, networks and storage devices.

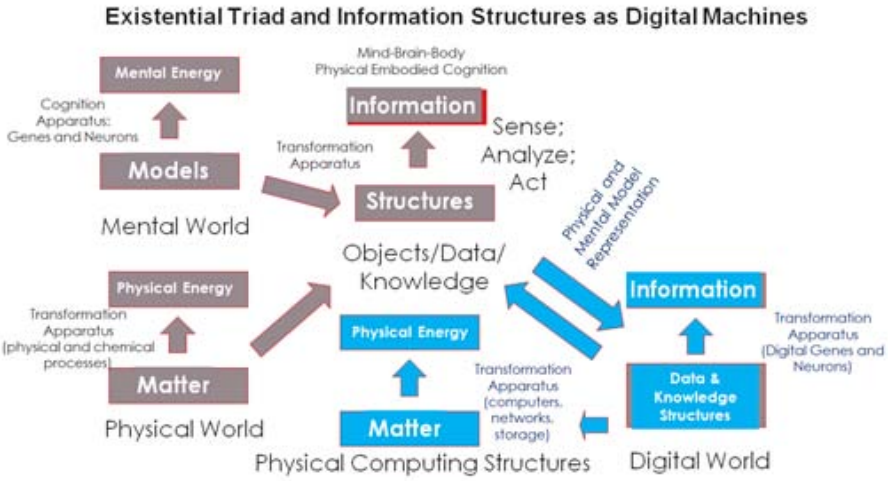

Figure 4. Digital Structures Modeling and Managing the Physical Structures.

The physical and cognitive structures are modeled as digital information processing structures and are used to establish sensors and actuators and manage the physical and mental worlds. Solutions of complex mathematical and practical problems are often using digital information processing. Current enterprise business process automation and the Internet based services have all been made possible using both the digital genes and neurons.

Next, we discuss the limitations of current information technologies and taking the cues from the embodied cognition in biological systems, we propose infusion of cognitive behavior in digital computing structures to implement self-managing patterns that provide resiliency and efficiency at scale. However, the limitation of current computing model stems from the inability of including the computing infrastructure itself in the model to sense and manage the physical world [16].

In addition to the self-referential circularity of the Turing computing model, the Church-Turing thesis boundaries were defied by non-stopping software and network systems such as operating systems and the Internet. These boundaries are even more challenged when rapid non-deterministic fluctuations drive the demand for resource readjustment in real-time without interrupting the service transactions in progress. This feature is more pronounced in the case of distributed computing structures that are composed of concurrent and asynchronous functions contributing to the interactions as in the case of business process automation tasks. The information processing structure in this case utilizes software components executed in hardware components from multiple infrastructure providers with local management systems enabling the deployment, operation and maintenance of the computation workloads on their infrastructure in the form of a cloud or a datacenter. The information processing system, in effect, behaves as complex autonomous system and is prone to emergent behavior in the face of strong fluctuations. For example, when the system is subject to sudden fluctuations in the demand for computing resources or sudden decrease due to failure of some components, the system will experience severe deviation from its mission unless the resources are restored

and the inconsistencies resulting from the local components being in different states are resolved.

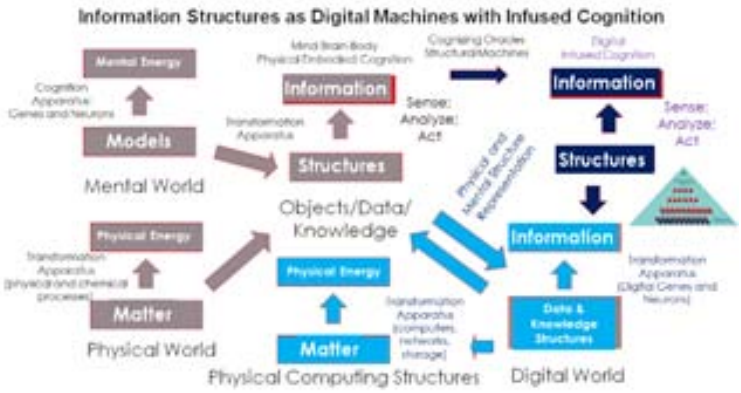

Figure 5. Digital Computing Structures with Infused Cognition.

Taking the cue from biological cognitive systems with self-managing patterns, we assert that the digital information processing structures must also become autonomous and predictive by extending their cognitive apparatus to include themselves and their behavior along with the information processing tasks at hand. Figure 5 shows the infusion of cognition into digital computing structures using the structural machine approach described in this paper.

Cognitive apparatuses that sense, model and monitor, reason and manage the digital computing structure allow encoding the information to deploy, monitor and manage the computing processes with local autonomy and global coordination. True intelligence involves generalizations from observations, creating models, deriving new insights from the models through reasoning. In addition, human intelligence also creates history and uses past behaviors and experience in making the decision. The cognitive overlay we propose in this paper provides a means to encode the information and the means to execute the processes required to understand the goals of the computational structure, available resources and the means to execute end to end deployment, monitoring and management to maintain homeostasis. In short, the new digital genome provides a means to create an autopoietic machine. In the next section, we discuss how the cognitive overlay allows the self-managing properties of auto-failover, auto-scaling and live migration of computing functions while maintaining local autonomy and global consistency. We demonstrate the implementation of such an autopoietic digital computing structure in an edge cloud.

7. Structural Machine Orchestrator (SMO): An Implementation of Triadic Automata with Managed Micro-Services Orchestration and Workload Mobility

We describe here a perspective implementation of the triadic structural machine, which processes its infware, software and hardware. We propose to use a cluster of edge clouds at Golden Gate University to demonstrate the autopoietic features by implementing the resiliency of a workload as a distributed web application that provides a specific service to many end points reached through the Internet. Figure 6 shows the proposed autopoietic machine configuration.

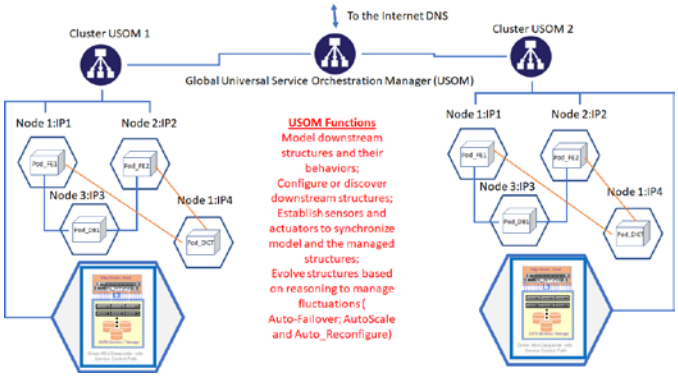

Figure 6. An Autopoietic Machine managing the resiliency of a cluster of edge clouds offering Kubernetes orchestrated workload (in the form of PoDs) connected to many endpoints through the Internet. USOM infuses cognitive control overlay network depicted by the hexagons. The data path connections are shown connecting the PoD network.

A high-performance edge cluster [51] is used as the hardware shown in figure 6. It provides low-latency computing as an edge cloud using commodity off-the-shelf computing hardware cluster connected at 100 GbE over fiber. It eliminates the complexity of legacy networking stacks by replacing merchant switch ASIC SDKs with a streamlined user space driver layer and using only L3 routing with leaf-spine topologies. In addition, it integrates low-latency storage (SSDs with RDMA and NVMeoF). The cluster management includes zero-touch provisioning, monitoring and run-time management of Infrastructure as a service (IaaS) to enable microservice or bare-metal workloads. The software consists of Kubernetes orchestration of a PoD network deployed in the IaaS which executes both the middleware (PaaS components such as the database and shared libraries/components) and specific application workloads and data. The particular workload shown here is a text spelling check application where various users create text which is checked in real-time for spelling accuracy.

The edge cluster proposed here is very typical of many cloud systems that offer IaaS and PaaS configuration and orchestration tools that are used to implement various workloads. However, the distinguishing and differentiating features of the triadic automata which infuse cognition into workload deployment, configuration, monitoring and self-management patterns are as follows:

1. A hierarchy of universal service orchestration managers (USOM) infuse cognitive overlay using the knowledge structures that model, configure or discover, monitor and manage both the physical and digital computing structures downstream using various sensors and actuators. The knowledge structures consist of the downstream entities with various attributes with internal relationships among them along with behaviors that will be executed whenever a state change occurs.
2. Each USOM is implemented as a structural machine executing the downstream knowledge structure evolution by synchronizing the model with physical and digital structures being managed using the sensors and actuators.
3. Changes in the state in the downstream structure invoke various behaviors that are encoded in the USOM to detect the change event, reason and determine appropriate action and act using the sensors and actuators that manage the downstream physical and digital structures.

8. Comparision with Current State of the Art and Conclusion

Current state of the art provides failover, scaling, and reconfiguration of workloads by using the automation of deploying, monitoring and reconfiguring either container-based or virtual machine-based images along with associated PoD and Virtual Machine networks. Workloads are locked within the IaaS and PaaS components and the automation of their management is performed by managing the IaaS and PaaS configurations. Figure 7 shows the current state of the art of workload deployment.

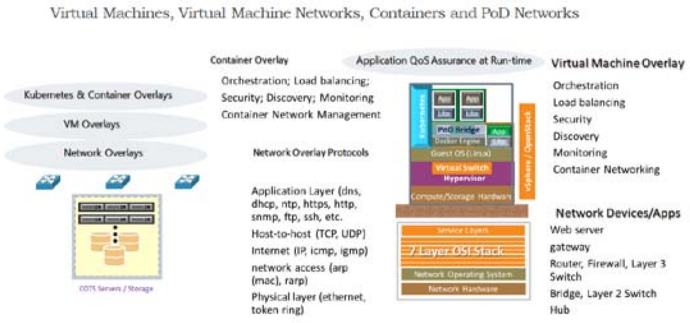

Figure 7. Current state of the art is a complex overlay of multiple layers evolved over a long period of time and are subject to the Church-Turing thesis boundaries in dealing with fluctuations in resource demand or availability.

The shortcomings of current state of the art and a potential new approach demonstrating workload mobility across multiple IaaS and PaaS infrastructures in multiple clouds without service disruption was published by Burgin and Mikkilineni in 2018 [26] based on the work carried out by one of the authors at C3DNA. In contrast to the current approach, that implementation was based on an extension to the Turing Machine by introducing a read/write instruction change based on an external cognizing oracle [52]. While this approach worked, it required an intervention in the data path and does not provide a generalized approach to infuse cognition with its modeling, reasoning and changing the evolution of the digital computing structure with systemic view.

The approach using triadic structural machines described in this paper provides a decoupling of workload or-

chestration from the Iaas and PaaS infrastructure orchestration. The cognitive layer models the IaaS and PaaS configurations, establishes sensors and actuators to synchronize the model with its physical structure and manages the evolution based on the cognitive knowledge infused in the knowledge structures. Figure 8 shows the new approach.

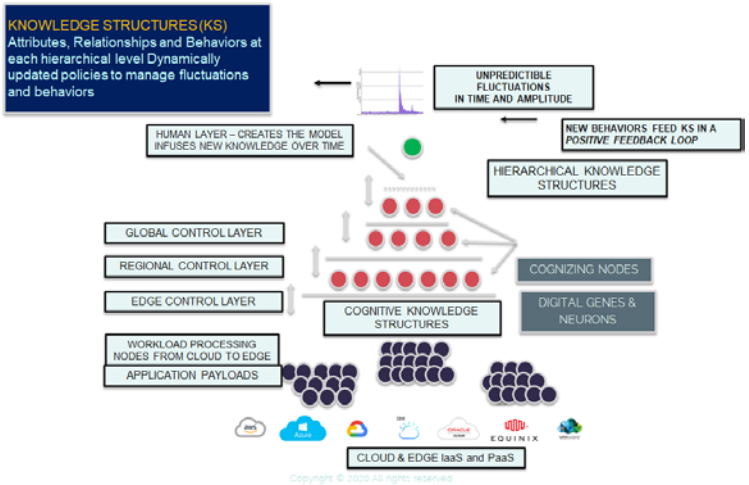

Figure 8. Workload orchestration using cognitive overlay in the form of knowledge structures, cognizing agents and structural machines.

The authors believe that the new mathematics and the proposed implementation of Triadic Automata will pave the way to acquire of a higher degree of efficiency, resiliency and scalability to digital information processing structures. First implementation of the Triadic Automata will be described in detail in a subsequent paper by the system developers using the edge clouds described above.

We conclude this paper with the observation made by von Neumann [53].

"It is very likely that on the basis of philosophy that every error has to be caught, explained, and corrected, a system of the complexity of the living organism would not last for a millisecond. Such a system is so integrated that it can operate across errors".

The cognitive layer infused in the Structural Triadic Automata provides the mechanism for the system to operate across errors in the face of nondeterministic fluctuations.

Acknowledgments

Rao Mikkilineni, acknowledges many valuable discussions with Gordana Dodig- Crnkovic, late Peter Wegner and Chip Ventors, who have contributed to his understanding of complex adaptive systems and their relevance to information technologies and business process automation.

Mark Burgin acknowledges his collaboration with Andrew Adamatzky, which was useful for the modeling of biological computing systems by structural machines. Collaboration with Narayan Debnath and Bidyut Gupta was helpful for the development of the theory of second-order algorithms.

References

[1] Maturana, Humberto R. & Varela, Francisco J. (1980). *Autopoiesis and Cognition*. The Realization of the Living. Dordrecht: Reidel.

[2] Yang, A., & Shan, Y. (eds.) (2008). *Intelligent Complex Adaptive Systems*, IGI Publishing, Hershey, PA.

[3] Arthur, W. B., Durlauf, S., & Lane, D. (eds.) (1997). *The Economy as an Evolving Complex System*. Addison-Wesley, Reading, MA.

[4] Dooley, K. (1997). A complex adaptive systems model of organizational change. Non-linear Dynamics, *Psychology and the Life Sciences*, v. 1, pp. 69-97.

[5] Choi, Thomas Y., Kevin J. Dooley, & Manus Rungtusanatham. (2001). Supply Networks and Complex Adaptive Sysems: Control Versus Emergence. *Journal of Operations Management*, v. 19, No. 3, pp. 351-66.

[6] Miller, J., & Page, S. (2007). *Complex adaptive Systems: An introduction to computational models of social life*. Princeton, NJ: Princeton University Press.

[7] Mitchell, M. (2009). *Complexity: a Guided Tour*. Oxford: Oxford University Press.

[8] Beinhocker, E. D. (2006). *The origins of wealth: evolution, complexity, and the radical remaking of economics*. Boston: Harvard Business School Press.

[9] Beinhocker, E. D. (2010). *Evolution as computation: Implications for economic theory and ontology*. Santa Fe Working Paper 2010-12037. Santa Fe: Santa Fe Institute.

[10] Waldrop, W. Mitchell. (1992). *Complexity: The Emerging Science at the Edge of Order and Chaos*. New York: Touchstone.

[11] Burgin, M. (2011). Information in the structure of the world. *International Journal Information Theories and Applications*, v. 18, No. 1, pp. 16-32.

[12] Prigogine, I. *Time, Structure and Fluctuations*. Available online: https://www.nobelprize.org/uploads/2018/06/prigogine-lecture.pdf (accessed on 16 June 2020).

[13] Prigogine, I., & Stengers, I. (1984). *Order out of Chaos*, Bantam Books, Toronto/New York/London.

[14] Burgin, Mark. (2017). The General Theory of Information as a Unifying Factor for Information Studies: The Noble Eight-Fold Path. *Proceedings*, v. 1, no. 3: 164.

[15] Frans de Waal. (2016). *Are We Smart Enough to Know How Smart Animals are?* W. W. Norton, New York.

[16] P. Cockshott, L. M. MacKenzie, & G. Michaelson. (2012). *Computation and Its Limits*, Oxford University Press, Oxford.

[17] R. Mikkilineni. (2012). Going beyond Computation and Its Limits: Injecting Cognition into Computing. *Applied Mathematics*, v. 3 No. 11A, pp. 1826-1835. doi: 10.4236/am.2012.331248.

[18] Burgin, M., & Adamatzky, A. (2017). Structural machines and slime mold computation. *Int. J. Gen. Syst*., v. 45, pp. 201-224.

[19] Burgin, M., & Adamatzky, A. (2017). Structural Machines as a Mathematical Model of Biological and Chemical Computers. *Theory Appl. Math. Comput. Sci*., v. 7, pp. 1-30.

[20] Burgin, M. (2020). Information Processing by Structural Machines, in Theoretical Information Studies: Information in the World, pp. 323-371.

[21] Burgin, M. (2017). Actors, Agents and Oracles in the Context of Artificial Intelligence. *Journal of Artificial Intelligence Research & Advances*, v. 4, No. 3, pp. 17-25.

[22] Rao Mikkilineni, Giovanni Morana., & Mark Burgin. (2015). Oracles in Software Networks: A New Scientific and Technological Approach to Designing Self-Managing Distributed Computing Processes, *Proceedings of the 2015 European Conference on Software Architecture Workshops* (ECSAW '15), 2015.

[23] Burgin, M. (2017). Inaccessible Information and the Mathematical Theory of Oracles, in *Information Studies and the Quest for Transdisciplinarity: Unity through Diversity*, World Scientific, New York/London/Singapore, pp. 59-114.

[24] Burgin, M. (2016). *Theory of Knowledge: Structures and Processes*. World Scientific Books: Singapore.

[25] Mikkilineni, R., & Burgin, M. (2020). Structural Machines as Unconventional Knowledge Processors. *Proceedings*, v. 47, 26.

[26] Burgin, M., & Mikkilineni, R. (2018). Cloud computing based on agent technology, super-recursive algorithms, and DNA. *Int. J. Grid and Utility Computing*, v. 9, No. 2, pp. 193-204.

[27] Turing, A. (1939). Systems of Logic Based on Ordinals, *Proc. Lond. Math. Soc.*, Ser.2, v. 45, pp. 161-228.

[28] Shettleworth, S. J. (2001). Animal cognition and animal behaviour. *Animal Behaviour*, 61: 277-286.

[29] Turing, A. (1936). On Computable Numbers with an Application to the Entscheidungs-problem, *Proc. Lond. Math. Soc.*, Ser. 2, v. 42, pp. 230-265.

[30] Von Neumann, J. (1949). Theory of Self-Reproducing Automata; University of Illinois Lectures on the Theory and Organization of Complicated Automata, Edited and completed by Arthur W. Burks; University of Illinois Press: Urbana, IL, USA.

[31] Burgin, M. (2005). *Superrecursive Algorithms*, Springer-Verlag, New York.

[32] Haykin, S. (1994). *Neural Networks: A Comprehensive Foundation*, New York, Macmillan.

[33] Kolmogorov, A. N. (1953). On the Concept of Algorithm, *Russian Mathematical Surveys*, v. 8, No. 4, pp. 175-176.

[34] Kleinberg, J., & Tardos, E. (2006). *Algorithm Design*, Pearson - Addison-Wesley.

[35] Goodrich, M. T., & Tamassia, R. (2015). *Algorithm Design and Applications*, Willey, Hoboken, NJ.

[36] Burgin, M. (2020). Triadic Automata and Machines as Information Transformers, *Information*, v. 11, No. 2, 102; doi: 10.3390/info11020102.

[37] Burgin, M. (2003). Nonlinear Phenomena in Spaces of Algorithms, *International Journal of Computer Mathematics*, v. 80, No. 12, pp. 1449-1476.

[38] Burgin, M. (2012). *Structural Reality*, Nova Science Publishers, New York.

[39] Robinson, A. (1963). *Introduction to Model Theory and Metamathematics of Algebra*, North-Holland, Amsterdam/New York.

[40] Yaglom, I. M. (1980). *Mathematical Structures and Mathematical Modeling*, Sov. Radio, Moscow (in Russian).

[41] Tegmark, M. (2008). The Mathematical Universe, *Foundations of Physics*, v. 38, No. 2, pp. 101-150.

[42] Bourbaki, N. (1957). *Structures*, Hermann, Paris.

[43] Bourbaki, N. (1960). *Theorie des Ensembles*, Hermann, Paris.

[44] C. Berge. (1973). Balanced hypergraphs and some applications to graph theory, in: J.N. Srivastava, ed., *A Survey of Combinatorial Theory* (North-Holland, Amsterdam, 1973) 15-23.

[45] Petri, C. A. (1962). *Kommunikation mit Automaten. English Translation, 1966: Communication with Automata*, Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York.

[46] Burgin, M., & Eberbach, E. (2009). On foundations of evolutionary computation: an evolutionary automata approach, in Hongwei Mo (Ed.), *Handbook of Research on Artificial Immune Systems and Natural Computing: Applying Complex Adaptive Technologies*, IGI Global, Hershey, Pennsylvania, pp. 342-360.

[47] Burgin, M., & Bratalskii, E. (1986). The principle of asymptotic uniformity in complex system modelling, in *Operation Research and Automated Control Systems*, Kiev: Institute of Cybernetics, pp. 115-122 [in Russian].

[48] Burgin, M., & Debnath, N. (2010). *Reusability as Design of Second-Level Algorithms*, in Proceedings of the ISCA 25th International Conference "*Computers and their Applications*" (CATA-2010), ISCA, Honolulu, Hawaii, 2010, pp. 147-152.

[49] Burgin, M., & Gupta, B. (2012). Second-level Algorithms, Superrecursivity, and Recovery Problem in Distributed Systems, *Theory of Computing Systems*, v. 50, No. 4, 2012, pp. 694-705.

[50] Burgin, M., Eberbach, E., & Mikkilineni, R. (2019). Cloud Computing and Cloud Automata as A New Paradigm for Computation. *Computer Reviews Journal*, v. 4, pp. 113-134. Retrieved from https://purkh.com/index.php /tocomp/article/view/459.

[51] R. Mikkilineni, & G. Morana. (2019). "Post-Turing Computing, Hierarchical Named Networks and a New Class of Edge Computing," 2019 *IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE), Napoli, Italy, pp. 82-87, doi: 10.1109/WETICE.2019.00024.

[52] Mikkilineni, R., Comparini, A., & Morana, G. (2012). The Turing o-machine and the DIME Network Architecture: Injecting the Architectural Resiliency into Distributed Computing. *Turing100, The Alan Turing Centenary*, EasyChair Proceedings in Computing. 2012. Available online: https://easychair.org/publications/paper/gBD (accessed on 12 May 2020).

[53] W. Aspray, & A. Burks. (1989). *Papers of John von Neumann on Computing and Computer Theory*. MIT Press, Cambridge.