Article

Infusing Autopoietic and Cognitive Behaviors into Digital Automata to Improve Their Sentience, Resilience, and Intelligence

Rao Mikkilineni

Ageno School of Business, Golden Gate University, San Francisco, CA 94105, USA; rmikkilineni@ggu.edu

Abstract: All living beings use autopoiesis and cognition to manage their "life" processes from birth through death. Autopoiesis enables them to use the specification in their genomes to instantiate themselves using matter and energy transformations. They reproduce, replicate, and manage their stability. Cognition allows them to process information into knowledge and use it to manage its interactions between various constituent parts within the system and its interaction with the environment. Currently, various attempts are underway to make modern computers mimic the resilience and intelligence of living beings using symbolic and sub-symbolic computing. We discuss here the limitations of classical computer science for implementing autopoietic and cognitive behaviors in digital machines. We propose a new architecture applying the general theory of information (GTI) and pave the path to make digital automata mimic living organisms by exhibiting autopoiesis and cognitive behaviors. The new science, based on GTI, asserts that information is a fundamental constituent of the physical world and that living beings convert information into knowledge using physical structures that use matter and energy. Our proposal uses the tools derived from GTI to provide a common knowledge representation from existing symbolic and sub-symbolic computing structures to implement autopoiesis and cognitive behaviors.

Keywords: artificial intelligence; symbolic computing; sub-symbolic computing; super-symbolic computing; knowledge networks; general theory of information

check for updates

Citation: Mikkilineni, R. Infusing Autopoietic and Cognitive Behaviors into Digital Automata to Improve Their Sentience, Resilience, and Intelligence. *Big Data Cogn. Comput.* **2022**, *6*, 7. https://doi.org/10.3390/bdcc6010007

Academic Editor: Tzung-Pei Hong

Received: 14 November 2021
Accepted: 31 December 2021
Published: 10 January 2022

1. Introduction

Our understanding of how living organisms manage their "life" processes from birth to death is throwing light onto how we can build digital machines that mimic them with the ability to manage their stability, safety, sustenance, and survival while fulfilling their goals even when external forces cause perturbations. As John von Neuman pointed out [1], there is a difference between machines and living organisms in how the stability of the system is maintained. Living organisms maintain their stability with self-knowledge, while the stability of digital machines (consisting of hardware and software components) has to be maintained through external means such as many operators and systems collaborating with each other.

All living beings, from single cells to human beings, possess two unique characteristics which allow them to manage themselves as autonomous systems with a purpose, manage their dynamics to maintain their stability, and achieve their goals. First, they are autopoietic [2], which means that they are made up of components and relationships which change over time without changing their identity as a system. They maintain equilibrium when external forces cause fluctuations in their internal or external interactions. They are continuously aware of their interactions within the components of the system and also of their interactions with their environment. Second, they are cognitive, which means that they know their own composition and the interactions, but also their interactions with their environment. They have apparatuses in the form of genes and neurons that enable structures for information processing in various forms to obtain knowledge of their dynamics and use it to manage their stability, safety, sustenance, and survival while achieving their goals.

In this paper, we examine how to model these behaviors using the tools made available through the general theory of information (GTI). We argue that we can use these models to infuse autopoietic and cognitive behaviors into digital automata to improve their sentience, resilience, and intelligence. Sentience involves information gathering and processing into knowledge. Resilience involves maintaining stability, and intelligence involves achieving the goals while managing safety, sustenance, and survival.

In essence, living systems convert information gathered through their senses into knowledge and use it to manage their stability, sustenance, safety, and survival. The knowledge obtained from different mechanisms is integrated with a common knowledge representation and is used to manage the system's structure and dynamics with autopoietic and cognitive behaviors. Therefore, if machines are to mimic living organisms, they must be infused with autopoietic and cognitive behaviors. This necessitates the need for tools that assist in self-organization and address self-referentiality for defining and executing autopoietic and cognitive behaviors. The concept of "self" and the inclusion of knowledge about the constituent components, their relationships, their interactions, and the consequent behavioral impact on the global state of the system plays a crucial role in infusing autopoietic and cognitive behaviors into current-generation digital computing machines.

Current state-of-the-art information technologies use symbolic (algorithms changing states of symbolic data structures) and sub-symbolic computations (neural network algorithms processing data). Symbolic computing has contributed to business process automation. Sub-symbolic computing has enabled computers to convert information from various sources (audio, text, images, and video) into knowledge and provide insights into hidden correlations. Current implementations of these two types of computations using John von Neumann's stored program implementation of the Turing machines are dependent on devices and methods that are governed by the Church–Turing thesis [3–5].

A very important use case utilizing symbolic and sub-symbolic computing is the automation of operation and management of a distributed business application, consisting of several software components, using computing resources made available by various cloud service providers on-demand in the form of infrastructure as a service (IaaS) and platform as a service (PaaS). A distributed application consists of multiple software components communicating with each other using computing (CPU and memory), networking, and storage resources. IaaS provides the required computing, networking, and storage resources. PaaS provides various middleware, tools, and libraries required to execute each component. Each software component contains the appropriate algorithms in an executable form and the data structures consisting of the state of the system at a particular instance of time on which the algorithm operates and evolves the computation. An application is thus a structure in the form of a complex multi-layer network consisting of various nodes executing certain functions and links that provide information exchange between the nodes. Each node performs specific functions based on its inputs from various links and provides output to various other components through various other links. Each component is made operational and managed by management systems and operators with local autonomy. This description of the application fits the description of a complex adaptive system (CAS). The dynamics and evolution in the time of a CAS depend very much on fluctuations in the system interaction within and with its environment. The structural stability of a CAS is dependent on its functions, structure, and fluctuations.

There are many successful approaches, tools, and processes widely discussed in the literature [6–11] to deploy, operate, and manage a distributed application's stability in the face of fluctuations in the demand for or the availability of resources or fluctuations caused by any external forces threatening the availability, performance, and security of the application during its execution. Symbolic computing allows the operation of distributed components using various autonomous, local IaaS, and PaaS management services. Sub-symbolic computing allows data gathered from various autonomous local IaaS and PaaS management systems to gain insights on the availability, performance, and security to manage the overall stability and safety of the application with the desired quality of service.

All the current state-of-the-art approaches use:

1. Either single vendor-provided tools augmented by home-grown approaches to tweak the availability, performance, and security management; or
2. Myriad third-party tools that manage multi-vendor cloud resources and provide end-to-end management of the distributed application components utilizing various local operation and management tools.

While both these approaches are utilized in managing application availability, performance, and security with varying levels of success, they have the following limitations:

1. The choice between the two approaches is between either single-vendor lock-in or the complexity of managing end-to-end application availability, performance, and security. While the single-vendor choice simplifies the operation and management using vendor-provided tools and services, migration from one vendor infrastructure to another vendor infrastructure is often expensive, time-consuming, and subject to sudden changes based on vendor policies, as some customers recently found out [12].
2. The current state of software development does not support creating composable workflows using multiple vendor-provided tools and software modules without increasing cost and complexity. It requires using the APIs provided by these vendors to build the end-to-end workflow, which results in increased complexity and effort in assuring the quality of the composed workflow.
3. As mentioned earlier, the distributed application, consisting of multiple components using resources managed by various local autonomous management systems operated and managed by multiple operators, acts as a complex adaptive system. Fluctuations in the resource demand or resource availability that impact a single component in the distributed system affect the overall application availability and performance. Local security breaches affect overall application security. Maintaining the stability and security of the application at desired levels of performance requires global knowledge sharing, coordination, and control while accommodating local autonomy and local constraints. This results in a "manager of managers" approach, where local management systems are managed by a global management system. This leads to the manager of managers conundrum, with the question "who manages the manager of managers?".
4. When the magnitude of fluctuations is small and the deviations are not far from the equilibrium values of availability, performance, and security, they could be corrected by making adjustments to the existing structure. However, if the magnitude of fluctuations is large enough to cause emergence properties of the CAS to appear, structural changes of the application are required (for example, auto-scaling, auto-failover, and live-migration of components) which would result in end-to-end service disruption during the period in which those structural changes are made. The larger the fluctuations, the shorter the time available for making the required changes without service disruption.
5. The knowledge obtained from sub-symbolic computing is hidden in the neural networks, and the insights derived have to be integrated with symbolic computing structures to implement the changes dictated by the insights, such as the structural changes required to maintain stability and security. In this paper, we discuss super-symbolic computation, which integrates the knowledge from symbolic and sub-symbolic computing using a common knowledge representation schema that surpasses the current data-structure-based schemas [13]. This approach is very similar to how the mammalian neocortex uses the reptilian brain to sense and process information.

In this paper, we discuss a new approach derived from the general theory of information [14–17] that infuses autopoietic and cognitive behaviors into application deployment, operation, and management processes. A super-symbolic computing architecture [18] is introduced with a common knowledge representation from the outputs of symbolic and sub-symbolic computing structures. A resulting knowledge network provides higher-level processes for organizing the constituent components into a system with self-regulation and maintaining stability in the face of fluctuations in its interactions within and with its environment outside.

In Section 2, we discuss the current state-of-the-art. In Section 3, we present GTI and the new science of information processing structures (SIPS) and provide the design for autopoietic and cognitive knowledge networks. In Section 4, we present a design where the new machines can improve the sentience, resilience, and intelligence of current state-of-the-art IT without disrupting their current implementations. We explore in Section 5, the consequences of this approach in the design of future information processing structures.

2. Current Computing Models

In this section, we will review current computing models and discuss why they are inadequate to incorporate autopoietic and cognitive behaviors in digital automata to mimic living organisms.

2.1. Symbolic Computing

All algorithms that are Turing computable fall within the boundaries of the Church–Turing thesis, which states that "a function on the natural numbers is computable by a human being following an algorithm, ignoring resource limitations, if and only if it is computable by a Turing machine" [3]. The resources here are the fuel for computation, including CPU and memory. The stored program implementation of the Turing machine provides a computing structure where the state is represented by symbols (1 s and 0 s) and an algorithm operates on them to change its state.

The concept of the universal Turing machine has allowed us to create general-purpose computers and [19] (p. 215) "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 a part of the world that includes themselves."

A closer examination of these observations points to two limitations for infusing autopoietic and cognitive behaviors and managing the structural stability of a distributed application under the influence of large fluctuations in its interactions using general-purpose computers. First, we cannot 'ignore resource limitations' when all systems have to manage finite resources and the fluctuations that impact them. Second, there is a logical limit to including the computer and the computed as a part of the total system with an identity and the knowledge about internal and external interactions incorporated in the computations.

Essential characteristics that autopoietic and cognitive systems exhibit are (i) a self-identity of the system that consists of several autonomous components with a variety of functions composed together as a structure; and (ii) the ability to receive information from the outside and inside of the system, integrate and interpret this information, and then activate a response. The system maintains a global model of the self and its interactions both inside and outside while utilizing local autonomy of constituents through embedded, embodied, extended, and enactive (4E) cognition. Autopoietic and cognitive behaviors are the organism's responses through natural evolution to manage stability in a CAS under the influence of fluctuations.

Why are these properties important? Two new drivers are pointing to the need for a new approach to using them:

1. Current business services demand non-stop operation, and their performance is adjusted in real-time to meet rapid fluctuations in service demand or available resources. A business application software providing these services contains many components which are often distributed across multiple geographies, and requires hardware (fuel for computation in the form of CPU and memory) owned or operated by different

providers managing infrastructure from multiple vendors. The current state-of-the-art requires a myriad of tools and processes to deploy and maintain the stability of the application. The result is complexity in managing local non-deterministic fluctuations in the demand for or the availability of the fuel impacting the end-to-end service quality. The solution today is either a single hardware infrastructure provider lock-in or the complexity of many third-party tools. Besides, the speed with which the quality of service has to be adjusted to meet the demand is becoming faster than the time it takes to orchestrate various infrastructure components such as virtual machine (VM) or container images, network plumbing, application configurations, middleware, etc. It takes time and effort to reconfigure distributed plumbing, which results in increased cost and complexity. The afore-mentioned Church–Turing thesis' boundaries are challenged when rapid non-deterministic fluctuations in the demand for or the availability of finite resources drive the need for structural readjustment in real-time without interrupting service transactions in progress. An autopoietic application would know to readjust the structure of the system ("self") with required resources and maintain its stability. This is analogous to how living organisms maintain homeostasis without having to interrupt critical functions.

2. Current business processes and their automation assume trusted relationships between the participants in various transactions. Information processing and communication structures in turn assume "trusted relationships" between their components. Unfortunately, global connectivity and non-deterministic fluctuations caused by participants make it necessary to verify trust before completing transactions. On the one hand, assuming a complete lack of trust between participants (whether they are people, service components, or devices) increases the effort to manage risk in every transaction to assure the safety and security of the transaction. On the other hand, assuming complete trust between the participants increases the security risk. What we need is a mechanism that trusts but verifies just as living beings do. They use cognition as a means to represent knowledge about the participants and the risk associated with them and use this knowledge in any interaction to manage the transactions. For example, a cognitive application would manage its own security and privacy, with appropriate knowledge, independent of hardware infrastructure security mechanisms using its own autopoietic behaviors. This is analogous to a living organism fighting harmful viruses with its own immune system. The knowledge of immune system processes is embedded as a part of the "Self" in the genome.

2.2. Sub-Symbolic Computing

Deep learning has delivered a variety of practical uses in the past decade. From revolutionizing customer experience, machine translation, language recognition, autonomous vehicle management, computer vision, text generation, speech understanding, and a multitude of other AI applications. This is very similar to how the reptilian brain uses 4E cognition to derive knowledge from the information provided by the five senses. However, the mammalian neocortex overlay uses a common knowledge representation across multiple inputs received from the five senses and uses it to execute autopoietic and cognitive behaviors. How the neocortex and the reptilian cortical columns process knowledge from the information gathered from the senses and representation of "self" and the interactions within and with the environment are well discussed by Yanai, and Martin, Stanislas Dehaene, Damasio, and Hawkins [20–24]. In the next section, we will discuss how GTI provides a model for a common knowledge representation from the information gathered via symbolic and sub-symbolic computing processes.

3. General Theory of Information and the Science of Information Processing Structures

GTI, structural machines, and various tools derived from GTI and the theory of structural machines dealing with the transformation of information and knowledge are discussed widely in several peer-reviewed journals, books, and other publications [14–17,25–33]. Although the theory has long been around, its application to model autopoietic and cognitive behaviors is just emerging [13,18,25,34].

In this section, we summarize the role of structural machines [15–17,25] in super-symbolic computation and their utility in infusing autopoietic and cognitive behaviors into digital automata. GTI explains and makes available constructive tools [26] (p. 4).

According to GTI, the material world consists of physical structures that are governed by the transformation laws of matter and energy. All physical and chemical structures contain potential information that characterizes their structure, the functions of their constituent parts when interacting with each other and with their surroundings, and their behaviors when internal and external factors cause fluctuations in their interactions. All material structures are complex adaptive systems with fluctuations driving their stability. If the fluctuations are small, they are in an equilibrium state with small perturbations from their mean behavior. If the perturbations are large, the structural stability is broken and the emergent properties dictate structural transformation. All knowledge about the structure, the functions, and the system's response to fluctuations is potential information and requires an observer to convert information into knowledge. Biological systems have evolved to build structures that sense the information from the material world and create mental structures that transform information into knowledge and use it to manage itself and its interactions with the external world to maintain stability in the face of fluctuations. GTI asserts that "knowledge to information is as the matter is to energy" and provides tools that convert information into knowledge that represents the state and the dynamics of the structure under consideration. We summarize here the basic concepts and tools from GTI which allow us to model autopoietic and cognitive behaviors and infuse them into digital automata.

3.1. Fundamental Triad or a Named Set

Structures in the material world exist subject to the laws of transformation of matter and energy. Living organisms have developed mental structures that receive information from the material world and convert it into knowledge using their physical structures. The knowledge gathered from their different information-processing structures is processed and used by the living organisms to exhibit both autopoietic and cognitive behaviors designed to manage their stability, safety, sustenance, and survival in interacting with the material world outside of them.

GTI provides a model for the knowledge representation process and the information and knowledge transformation processes. The elemental structure is called a "fundamental triad or a named set" [25] (p. 3). It has the following visual representation shown in Figure 1.

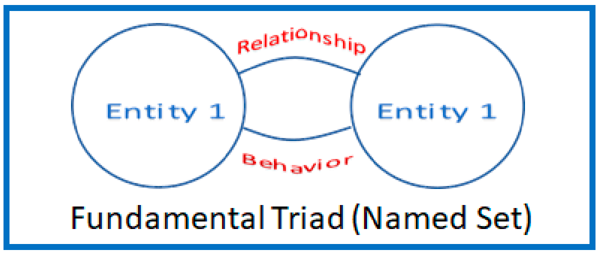

Figure 1. A named set or a fundamental triad can represent epistemic connections between two entities as a knowledge structure derived from information.

Every entity has a name. An entity with its own name is connected to another entity. The connection, which itself has a name, depicts the knowledge of the relationship between the entities. The behavioral evolution, when a state change occurs in either object, is represented by an action that results in their change in state. The behavioral change could be either local within the object or through induced behavior using information exchange among the entities based on their relationship. Entities wired together fire together to execute the collective behavior. Unlike a data structure, the named set incorporates entities, relationships, and behaviors. The named sets are the components of a knowledge structure.

It is demonstrated that fundamental triads, i.e., named sets, form the most basic structures in nature, technology, and society—in fact, in everything. This means that "the general theory of information provides means for a synthesis of physics, psychology and information science playing the role of a metatheory for these scientific areas. At the same time, the new definition characterizes proper information when the general concept is specified by additional principles. The construction of an infological system allows researchers to exactly delineate information in the area of their studies." [26] (p. 4).

3.2. Knowledge Structure

A knowledge structure [25,27–29] is composed of related fundamental triads, and any state change causes behavioral evolution based on connections and relationships. The theory of knowledge states that information is transformed into knowledge in the form of fundamental triads and their composite structures. Any changes in information cause corresponding changes in knowledge structures based on their relationships and associated behaviors defining the state and its dynamics. Knowledge processing is an important feature of intelligence in general and artificial intelligence in particular. To develop computing systems working with knowledge, it is necessary to elaborate the means of working with knowledge representations (as opposed to data and symbolic data structures) because knowledge is an abstract structure. It is important to emphasize the differences between data, data structures, knowledge, and knowledge structures. Data are mental or physical "observables" represented as symbols.

Data structures define patterns of the relationships between data items. Knowledge is a system that includes data enhanced by relations between the data *and* their behaviors when data change (with new information) occurs in the represented domain. Knowledge structures include data structures abstracted to various objects, as well as their inter-object, and intra-object relationships and behaviors. The corresponding state vector defines a named set of knowledge structures, a representation of which is illustrated in Figure 2. When an event occurs causing changes to the objects, the new information is processed and the knowledge structures are updated based on their relationships and behaviors.

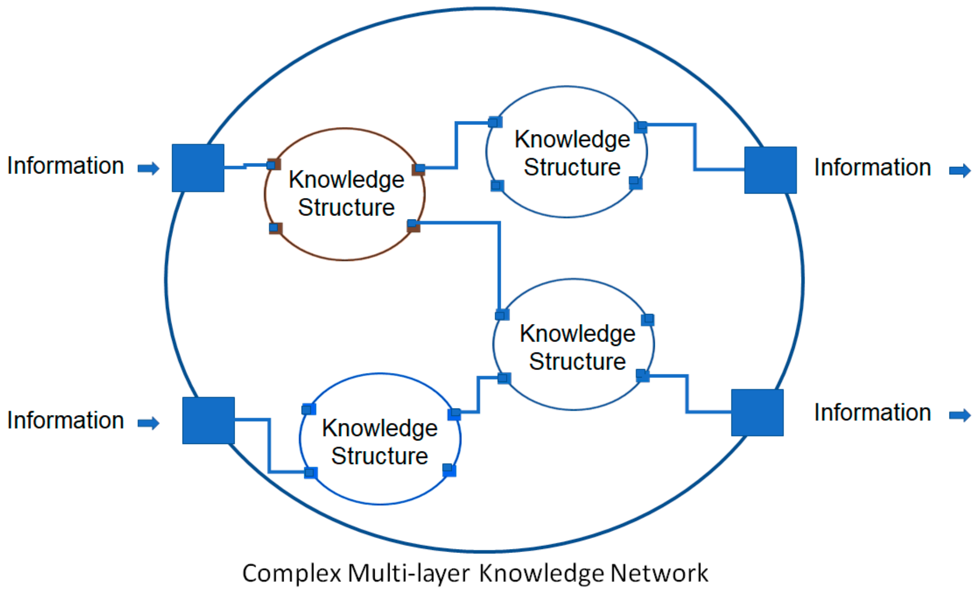

Figure 2. A knowledge network specifying the knowledge structures where information is communicated between nodes through barrios links. The nodes define the structure and function of the domain knowledge in the form of named sets.

The knowledge structure thus represents the system state and its evolution as a complex multi-layer network where the wired nodes fire together to execute global behavior. The knowledge structure in each node maintains its state and evolves using conventional processors. Any information change produces a functional behavior change in the knowledge structure node.

3.3. Structural Machine

A structural machine is a tool derived from GTI. Just as an algorithm operates on the symbolic data structure in a Turing machine, an algorithm operates on the knowledge structure in a structural machine [25]. A knowledge structure consists of a network (as discussed earlier) with nodes defining local behaviors and links defining the relationships between nodes that exchange information. When the exchanged information causes an event (such as a change in the value of an attribute in the knowledge structure), the structural machine executes the algorithm to change the knowledge structure from one state to another, as shown in Figure 3. The structural machine is implemented using multiple processors, also shown in Figure 3. The details of how to represent knowledge in the knowledge network schema, how to create various instances of the knowledge network, and how to execute the dynamics of the knowledge network using various operations on the knowledge network schema are detailed in multiple publications [15–17]. We refer the reader for details of the implementation using general-purpose computing structures including cloud computing services. Suffice it to say that structural machines operate on knowledge structures, whereas Turing machines operate on data structures.

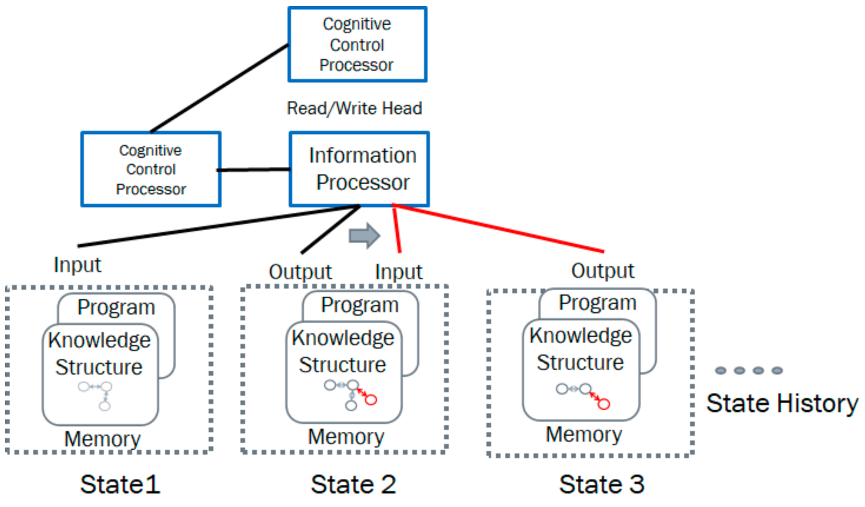

Figure 3. A structural machine operating on knowledge structure schema.

In summary, a structural machine [25] is an abstract automaton that works with structures of a given type and has three components:

1. The unified control device CM regulates the state of the machine.
2. The unified processor performs the transformation of the processed structures and their actions (operations) depending on the state of the machine and the state of the processed structures.
3. The functional space, which in turn consists of three components:
   - The input space which contains the input knowledge structure.
   - The output space which contains the output knowledge structure.
   - The processing space in which the input knowledge structure is transformed into the output structure.

3.4. Oracles and Self-Managing Computing Structures

Living organisms are self-organizing and self-managing systems with autopoietic and cognitive behaviors. Self-organization and management involve the knowledge of the "Self" as a system, and also the knowledge of the interactions within the components of the system and the system's interactions with its environment. It also implies that the system is aware of its goals and has the ability to monitor and manage its interactions to achieve those goals. The concepts of self and self-reflection that model the observer, the observed, and their interactions are embedded in the genome specification [20–24] in the form of autopoietic and cognitive behaviors.

The oracles in software networks [30], an extended concept of a Turing oracle introduced in the Turing O-Machine [5,32] in the context of the theory of oracles, provide a means to introduce both autopoietic and cognitive behaviors into the structural machines, going beyond the limitation of the Turing machine-based general-purpose computers mentioned earlier [19] (p. 215).

The controller in the structural machine allows implementing a cognizing agent or the oracle overlay that manages the downstream processors and associated knowledge structure evolution. According to the theory of oracles [30], an agent system can work as an oracle, collecting information for the basic machines or more complex information processing devices such as a network or computer cluster. For instance, in the theory of super-recursive algorithms and hypercomputation [31,33], computation, an important type of oracle, contains information on whether a Turing machine halts when it is given a definite input or not. As we know in this case, even simple inductive Turing machines [33] can compute, i.e., obtain this and other information that is incomputable, i.e., unreachable, by Turing machines. Consequently, it is possible to use inductive Turing machines as oracles for bringing information that is not computable by Turing machines and other recursive algorithm information to various computing devices such as Turing machines or neural networks. In essence, the oracles provide a means to abstract knowledge in the form of a state and for its evolution based on the information provided by various other sources such as symbolic computing, sub-symbolic computing structures, common sense, and other domain-specific ontologies, along with their behaviors.

3.5. Triadic Automata and Autopoietic Machines with Cognitive Behaviors

Current-day digital information processing structures include hardware that provides the resources for computation (CPU, memory) and software that contains the algorithm in the form of a program that operates on data structures. The data structures represent the state of the system. The stability of the computing structures and the evolution of the state depends on the operation and management of both the hardware and software components. Triadic automata [15–17,25] provide the means to include hardware and software behaviors in the knowledge representation to evolve the system as a unit by including a new component called infware [33]. The "life" process dynamics of managing software and hardware constitute the software. The picture shows the three-level structures that model the life processes of both the computer and the computed to provide the system's autopoietic and cognitive behaviors.

Physical devices contain the hardware structures (computers, network, storage, sensors, actuators, etc.). Physical devices contain the hardware structures (computers, network, storage, sensors, actuators, etc.) Process management and control provide the evolution of the physical structures. The software contains algorithms, programs, and applications. Process management and control provide the evolution of the software structures. The software contains data structures, information structures, and knowledge structures. Process management and control provide the evolution of the structures based on the interactions within the structures and with their environment. Figure 4 shows the triadic automata.

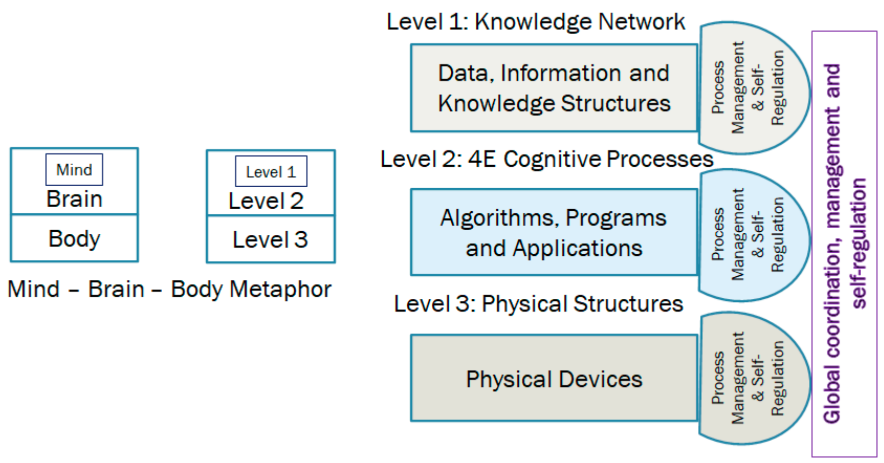
Mind
Brain
Body
Level 1
Level 2
Level 3
Mind – Brain – Body Metaphor
Level 1: Knowledge Network
Data, Information and Knowledge Structures
Level 2: 4E Cognitive Processes
Algorithms, Programs and Applications
Level 3: Physical Structures
Physical Devices
Process Management & Self-Regulation
Global coordination, management and self-regulation

Figure 4. Triadic automata managing hardware, software, and "infware".

In the next section, we discuss how to implement the structural machine using current state-of-the-art symbolic and sub-symbolic computations and discuss two use cases that exploit the autopoietic and cognitive behaviors.

4. Infusing Autopoietic and Cognitive Behaviors into Existing Information Technology Use Cases

GTI tells us [28] that information in the physical world is transformed into knowledge using abstract mental structures by living organisms, and they use this knowledge to exhibit autopoietic and cognitive behaviors building symbolic and sub-symbolic computing structures which exploit the physical and chemical processes converting matter and energy. In addition, it helps us define [34] the digital genome as a collection of knowledge structures [13–15] coded in an executable form to be processed with "structural machines" [25] implemented using digital genes (symbolic computing structure) and digital neurons (sub-symbolic computing structure). "The digital genome enables digital process execution to discover the computing resources in the environment, use them to assemble the hardware, cognitive apparatuses in the form of digital genes and digital neurons and evolve the process of sentient, resilient, intelligent and efficient management of both the self and the environment with embedded, embodied, enacted and extended (4E) cognitive processes. The digital genome incorporates the knowledge in the form of hierarchical intelligence with a definition of the sentient digital computing structures that discover, monitor, and evolve both the self and the interactions with each other and the environment based on best practices infused in them. The digital genome specifies the execution of knowledge networks using both symbolic computing and sub-symbolic computing structures. The knowledge network consists of a super-symbolic network of symbolic and sub-symbolic networks executing the functions defined in their components. The structure provides the system's behavior and evolution, maintaining the system's stability in the face of fluctuations in both internal and external interactions. The digital genome encapsulates both autopoietic and cognitive behaviors of digital information processing structure capable sentience, resilience and intelligence" [34] (p. 3).

In this section, we propose the design of a digital genome to infuse autopoiesis and cognition into an application deployed in a public cloud using locally available infrastructure as a service (IaaS) and platform as a service (PaaS). IaaS provides the CPU and memory as the fuel for executing the software algorithm. It is usually provisioned using computing hardware and operating systems, which provide the required resources that are managed to meet appropriate availability, security, performance, and other non-functional requirements. PaaS provides the middleware and other tools required to manage the data structures that the algorithm operates on. These include components such as databases, file systems, web servers, application servers, etc., and the associated libraries.

Figure 5 shows various components that are required to infuse autopoietic and cognitive behaviors, which provide the application with self-regulation capabilities, with a sense of self-awareness, and the knowledge to obtain resources required from various sources and use them to deploy, operate and manage both the self (the distributed application) and the resources to manage stability and successful execution of the application intent.

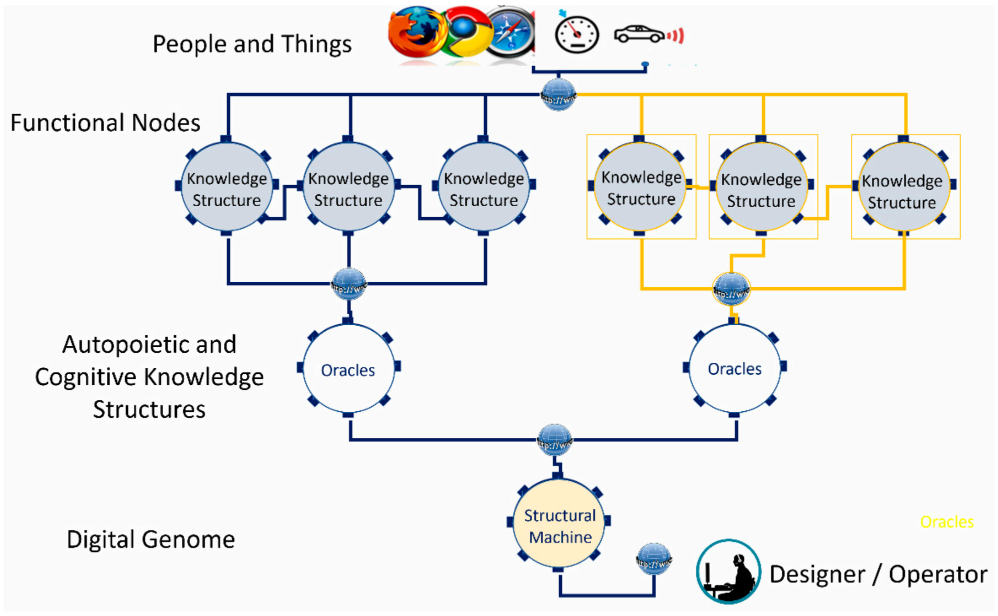

Figure 5. Knowledge network where each node performs several functions based on inputs received from various ports and communicates information to other nodes through output ports. The nodes wired together fire together to execute collective behaviors.

Just as the genome in living organisms contains the information required to manage life processes [20], the digital genome contains all the information in the form of knowledge structures to build itself, reproduce itself, and maintain its stability while using its cognitive processes to interact with the outside world, whether this is the hardware infrastructure it uses to build itself or the business of connecting people, things, and business services. The input to the genome is the information about where to obtain resources and the software components. It also includes information about "life" processes in the form of knowledge structures as well as autopoietic and cognitive behaviors to build itself, replicate, and deploy various components and manage their stability.

We depict here a functional node that uses cloud IaaS services to create the fuel for computation, and PaaS services for providing the software components required for the application workloads to operate on the data, resulting in a change in the state of the system. Here we show the containers and their management to provide the local management of the IaaS, PaaS, and the application workloads, as shown in the functional node and the digital node.

The triadic automata described in [15,17] are implemented using a structural machine composed of five components:

1. **The Digital Genome Node:** Contains both the cognitive "reasoning" processes and the autopoietic "life" processes to configure, monitor, and manage the downstream resources (e.g., IaaS, PaaS, and application component workloads, which include both programs in executable form and associated data). Operations defined [15,17] in the digital genome provide the required behaviors, such as replication, transcription, and translation, to deploy appropriate downstream networks and their component functions just as cellular organisms do. By "life" processes we mean the structure, its initial state information, knowledge about resources, and processes to configure, monitor, and manage based on known-best-practice knowledge encapsulated in exe-

cutable form. Examples are executable programs to configure, monitor, and manage IaaS, PaaS resources, and application business process executables as workloads and associated data using local resource managers provided by multiple vendors.

2. **Autopoietic Network Node**: Contains knowledge about the autopoietic "life" processes that deal with configuring, monitoring, and managing the resources using replication, transcription of policies, and best practices to execute them. The autopoietic network node contains the knowledge structures that configure, monitor, and manage downstream autopoietic function nodes that are designed to configure, monitor, and manage the local resources.
3. **Autopoietic Function Node**: Contains knowledge about the autopoietic "life" processes that deal with configuring, monitoring, and managing the local resources using replication, transcription of policies, and best practices to execute them. For example, if the resources are available in a public cloud, the autopoietic node deploys the executable programs that use local management programs such as Kubernetes, and containers or scripts that provision, monitor, and manage the resources. The IaaS autopoietic managers configure, monitor, and manage the cloud IaaS services using various scripts or tools. The PaaS autopoietic manager configures, monitors, and manages PaaS services in the local environment. For example, Dockers and Kubernetes are configured and managed in the local environment using local scripts or tools. The knowledge structures in these nodes are designed to capture local resources and manage them. Databases, web servers, and other middleware are examples of PaaS managed by the autopoietic PaaS function node.
4. **Cognitive Network Node**: Contains knowledge about the application-specific "life" processes that deal with configuring, monitoring, and managing the application workloads and associated data using replication, transcription of policies, and best practices to execute them. The cognitive network node contains the knowledge structures that configure, monitor, and manage downstream application components as cognitive function nodes that are designed to configure, monitor, and manage the application executables and associated data. For example, the cognitive function has the knowledge about the components of the application such as the web component, application logic component, and database component, as well as where to deploy, monitor, and manage them based on best-practice policies when fluctuations point to potential instabilities. When a local component fails, the cognitive function node has the knowledge to recover with auto-failover or to configure based on recovery position objective (RPO) and recovery time objective (RTO), or to recover without service description or auto-scale to maintain stability using replication and managing the data to maintain service while replicating, etc.
5. **Cognitive Function Node**: Contains knowledge about configuring, monitoring, and managing the local application function using the PaaS and IaaS resources made available with autopoietic function nodes.

Autopoietic and cognitive processes encapsulate the knowledge about an application, create a sense of "self", and select and use the "life" processes encoded and made available in the digital genome to maintain stability while the application interacts with the environment (users, devices, other applications, etc.). Figure 6 shows how the digital genome, the knowledge network managers, and functional nodes could be implemented using IaaS and PaaS resources from cloud providers to create a self-regulating cognitive distributed application.

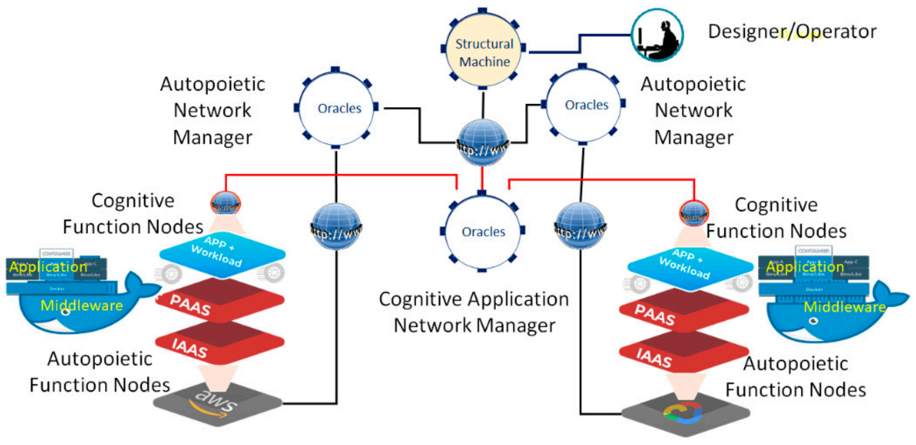

Figure 6. A self-regulating application implemented as a knowledge network managed by autopoietic and cognitive network regulators. (See the video for how autopoiesis and cognition improve sentience, resilience, and intelligence of an application (https://youtu.be/WS6cfFN4X3A accessed on 30 December 2021).

Figure 6 shows a potential implementation of a self-regulating application using IaaS and PaaS services in two different clouds. The autopoietic functional nodes provide the processes required to utilize local IaaS and PaaS structures. The stability of the structure is managed by the autopoietic network regulators. The cognitive function nodes provide the means for the application workload and data to be configured, monitored, and managed based on cognitive network regulators.

In summary, the digital genome specifies the "life" processes of a distributed application where the functions, structure, and means for dealing with fluctuations are encoded. In addition, knowledge about various resources and their management using local autonomous management systems is used by the autopoietic "life" processes encoded in the digital genome. An instance of the digital genome is executed using a structural machine that instantiates, replicates, and specializes the knowledge network managers and various functional nodes. The autopoietic functional nodes manage the local resources and are orchestrated by the autopoietic network managers. The application workloads are orchestrated by the cognitive application nodes and their orchestrators. This approach allows both autopoietic and cognitive behaviors to be orchestrated system-wide using a global knowledge network.

It is important to note that the digital genome's purpose is not to manage any particular cloud IaaS and PaaS. It is to understand the purpose of the distributed application and specify its execution and evolution utilizing various distributed resources available (cloud or no cloud). It also has the knowledge to manage its safety and stability during its execution in the face of large fluctuations that cause deviations from its structural equilibrium. If the fluctuations are large and have the potential to cause disruption, then the system will use its cognitive knowledge-based reasoning and reconfigure itself using the processes encapsulated in the autopoietic and cognitive regulators.

5. Conclusions

Biological organisms have exploited autopoietic and cognitive behaviors to improve their sentience, resilience, and intelligence. Recent studies in neuroscience, cell biology, and genomics have thrown more light on how humans harness information and knowledge with "self" awareness and awareness of their interactions with each other and their environment. GTI enables us to model autopoietic and cognitive behaviors and infuse them into digital automata.

Alan Turing's observation about how humans count numbers led to the concept of the universal Turing machine. John von Neumann demonstrated the utility of the Turing machine with his stored program implementation. Maxwell's equations described the fundamental relationship between electricity, magnetism, and wave propagation. Hertz discovered radio waves and Marconi built a radio device.

GTI and the tools provided by Prof. Mark Burgin pave the path for infusing autopoietic and cognitive behaviors into digital machines with a model of how information and knowledge transformations using physical structures provide a framework for modeling and improving a system's resilience and intelligence. Only time will tell if the model of the knowledge network where the functional nodes wired together fire together to exhibit collective behavior provides a robust framework for building intelligent machines that mimic biological organisms.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: The author wishes to thank Mark Burgin for various in-depth discussions on the general theory of information and the theory of structural machines.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Von Neumann, J. *Theory of Self-Reproducing Automata*; Burks, A.W., Ed.; University of Illinois Press: Urbana, IL, USA, 1966; p. 71.
2. Luisi, P.L. Autopoiesis: A review and a reappraisal. *Naturwissenschaften* **2003**, *90*, 49–59. [CrossRef] [PubMed]
3. Copeland, B.J. The Church-Turing Thesis. In *The Stanford Encyclopedia of Philosophy*; Edward, N.Z., Ed.; Stanford University: Stanford, CA, USA, 2020. Available online: https://plato.stanford.edu/archives/sum2020/entries/church-turing/ (accessed on 30 December 2021).
4. Church, A. A note on the Entscheidungsproblem. *J. Symb. Log.* **1936**, *1*, 40–41. [CrossRef]
5. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. *Proc. Lond. Math. Soc.* **1937**, *2*, 230–265.
6. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
7. Buyya, R.; Yeo, C.S.; Venugopal, S.; Broberg, J.; Brandic, I. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **2009**, *25*, 599–616. [CrossRef]
8. ANunet. Whitepaper: A Global Decentralized Framework. Available online: https://nunet-io.github.io/public/NuNet_Whitepaper.pdf (accessed on 30 December 2021).
9. Mao, H.; Schwarzkopf, M.; Venkatakrishnan, S.B.; Meng, Z.; Alizadeh, M. Learning scheduling algorithms for data processing clusters. In Proceedings of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 21–25 August 2017; ACM Press: New York, NY, USA, 2019; pp. 270–288.
10. Ghafouri, S.; Saleh-Bigdeli, A.A.; Doyle, J. Consolidation of Services in Mobile Edge Clouds using a Learning-based Framework. In Proceedings of the 2020 IEEE World Congress on Services (IEEE SERVICES 2020), Beijing, China, 7–11 July 2020; IEEE: New York, NY, USA, 2020; pp. 116–121.
11. Brandherm, F.; Wang, L.; Mühlhäuser, M. A learning-based framework for optimizing service migration in mobile edge clouds. In Proceedings of the 2nd International Workshop on Edge Systems, Analytics, and Networking, New York, NY, USA, 25 March 2019; pp. 12–17.
12. Cut Off from the Cloud: How Free Speech-Touting Parler Became a Tech Industry Pariah. 13 January 2021. Available online: computerweekly.com (accessed on 30 December 2021).
13. Burgin, M.; Mikkilineni, R. From Symbolic Computation to Super-Symbolic Computation, EasyChair Preprint No. 6559. 2021. Available online: https://easychair.org/publications/preprint_open/PMjC (accessed on 30 December 2021).
14. Burgin, M. *Theory of Information: Fundamentality, Diversity, and Unification*; World Scientific: Singapore, 2010.
15. Burgin, M. Triadic Automata and Machines as Information Transformers. *Information* **2020**, *11*, 102. [CrossRef]
16. Burgin, M.; Mikkilineni, R.; Phalke, V. Autopoietic Computing Systems and Triadic Automata: The Theory and Practice. *Adv. Comput. Commun.* **2020**, *1*, 16–35. [CrossRef]
17. Burgin, M.; Mikkilineni, R. From Data Processing to Knowledge Processing: Working with Operational Schemas by Autopoietic Machines. *Big Data Cogn. Comput.* **2021**, *5*, 13. [CrossRef]
18. Burgin, M.; Mikkilineni, R. Symbiotic information processing: A methodological analysis. In Proceedings of the Theoretical and Foundational Problems in Information Studies (TFP), is4si Summit 2021, Online, 12–19 September 2021.
19. Cockshott, P.; MacKenzie, L.M.; Michaelson, G. *Computation and Its Limits*; Oxford University Press: New York, NY, USA, 2012; p. 215.
20. Yanai, I.; Martin, L. *The Society of Genes*; Harvard University Press: Cambridge, MA, USA, 2016.
21. Stanislas, D. *Consciousness and the Brain: Deciphering How the Brain Codes Our Thoughts*; Tantor Audio: New York, NY, USA, 2014.

22. Stanislas, D. *Reading in the Brain: The New Science of How We Read Revised and Updated*; Penguin Books: New York, NY, USA, 2010; p. 10.
23. Damasio, A. *Self Comes to Mind. Pantheon, a Division of Random House*; Pantheon: New York, NY, USA, 2010.
24. Hawkins, J. *A Thousand Brains: A New Theory of Intelligence*; Basic Books: New York, NY, USA, 2021.
25. Mikkilineni, R.; Burgin, M. Structural Machines as Unconventional Knowledge Processors. *Proceedings* **2020**, *47*, 26. [CrossRef]
26. Burgin, M. The General Theory of Information as a Unifying Factor for Information Studies: The Noble Eight-Fold Path. *Proceedings* **2017**, *1*, 164. [CrossRef]
27. Burgin, M. *Theory of Named Sets*; Nova Science Publisher Inc.: New York, NY, USA, 2011.
28. Burgin, M. *Theory of Knowledge: Structures and Processes*; World Scientific Books: Singapore, 2016.
29. Burgin, M. Data, Information, and Knowledge. *Information* **2004**, *7*, 47–57.
30. Mikkilineni, R.; Morana, G.; Burgin, M. Oracles in Software Networks: A New Scientific and Technological Approach to Designing Self-Managing Distributed Computing Processes. In Proceedings of the 2015 European Conference on Software Architecture Workshops (ECSAW '15), Croatia, Balkans, 7–11 September 2015; ACM: New York, NY, USA, 2015.
31. Burgin, M.; Mikkilineni, R. Cloud computing based on agent technology, superrecursive algorithms, and DNA. *Int. J. Grid Util. Comput.* **2018**, *9*, 193–204. [CrossRef]
32. Mikkilineni, R.; Comparini, A.; Morana, G. 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 21 December 2021).
33. Burgin, M. *Superecursive Algorithms*; Springer: New York, NY, USA, 2005.
34. Burgin, M.; Mikkilineni, R. On the Autopoietic and Cognitive Behavior, EasyChair Preprint No. 6261, Version 2. 2021. Available online: https://easychair.org/publications/preprint_open/tkjk (accessed on 21 December 2021).