DOI 10.1007/s10559-020-00266-1
Cybernetics and Systems Analysis, Vol. 56, No. 3, May, 2020

NEW MEANS OF CYBERNETICS, INFORMATICS, COMPUTER ENGINEERING, AND SYSTEMS ANALYSIS

CHAOTIC ARCHITECTURES: A NEW TREND IN COMPUTERS

A. V. Palagin,[1†] **M. V. Semotiuk,**[1‡] **and S. V. Ustenko**[2]

UDC 004

Abstract. *Information technologies are analyzed and their components are identified as virtualization technologies, quantitative technologies, data technologies, and knowledge technologies. Based on the analysis, it is determined that chaotic architectures of computer systems are a new trend in the development of these systems.*

Keywords: *information technology, technology levels, virtualization technology, data technology, knowledge technology, chaos, chaotic architectures, smart systems, computers, programming paradigm, machine algebra.*

INTRODUCTION

At a present time, we use such words and concepts as “information,” “information technology,” “information data,” etc., not thinking about thousand years of history of formation and development of information technology (IT). It is courtesy to it that we were able to preserve the history of mankind and its cultural legacy accumulated as a result of succession of generations constantly replenishing it with new information content. Most often, IT is associated precisely with computer technology, and for a good reason: the appearance of computers brought this technology to a new modern level. Since information interaction is a complex of symmetrical interactions, the following question arises: what kind of influence does modern IT have on the development of computing technologies?

LEVELS OF MODERN INFORMATION TECHNOLOGY

Systemic and technical foundations of creating computing technologies determine the essential features and characteristics of this class of technology determined by the current capabilities of IT. These features and qualities determine the functional level of modern technology. In a broad sense, any technology is an amount of knowledge and tools that can be used to produce goods and services as economic resources. In the field of computing technologies, information technology is certainly instrumental. Figure 1 shows the components of IT, i.e., four levels of technology: knowledge technology (KT), data technology (DT), quantitative technology (QT), and virtualization technology (VT).

The process of formation of information technology has developed historically and began with virtualization technology, when a certain group of objects was assigned a value of quantitative indexes. At the same time, not the objects were evaluated as such, but their virtual images. Thus, virtualization technology is a class of technology that allows us to perceive environmental objects through quantitative characteristics usually in an impersonal form, and is instrumental for information technology as a whole.

[1]V. M. Glushkov Institute of Cybernetics, National Academy of Sciences of Ukraine, Kyiv, Ukraine, [†]*palagin_a@ukr.net*; [‡]*semo@i.ua*. [2]Kyiv National Economic University named after Vadym Hetman, Kyiv, Ukraine, *stasustenko@ukr.net*. Translated from Kibernetika i Sistemnyi Analiz, No. 3, May–June, 2020, pp. 184–193. Original article submitted September 05, 2019.

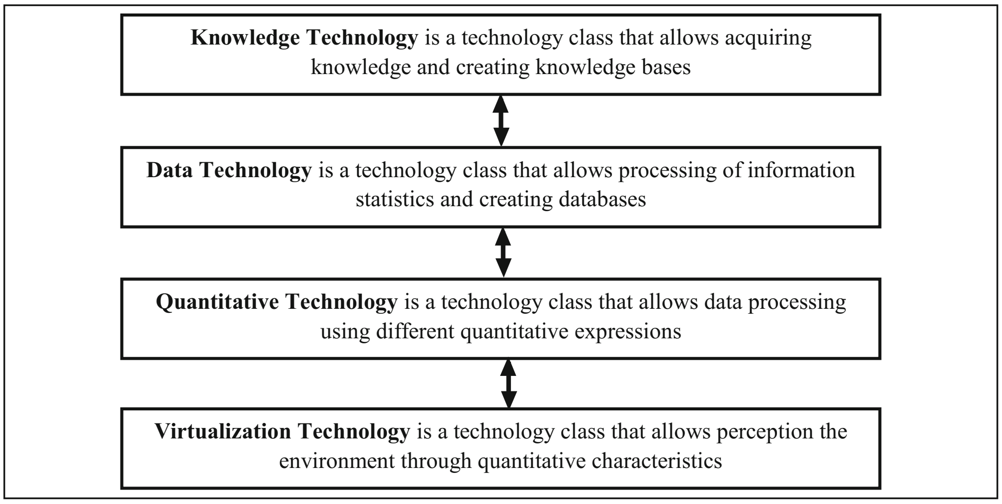
Knowledge Technology is a technology class that allows acquiring knowledge and creating knowledge bases
Data Technology is a technology class that allows processing of information statistics and creating databases
Quantitative Technology is a technology class that allows data processing using different quantitative expressions
Virtualization Technology is a technology class that allows perception the environment through quantitative characteristics

Fig. 1. Levels of modern information technology.

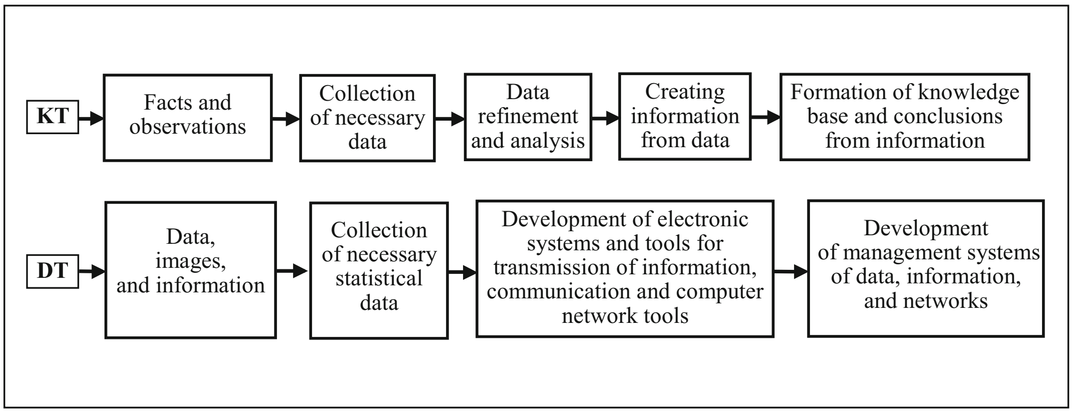
KT
Facts and observations
Collection of necessary data
Data refinement and analysis
Creating information from data
Formation of knowledge base and conclusions from information
DT
Data, images, and information
Collection of necessary statistical data
Development of electronic systems and tools for transmission of information, communication and computer network tools
Development of management systems of data, information, and networks

Fig. 2. Structure of upper levels of IT.

The second level of IT is represented by quantitative technology. They allow us to process information about certain objects in an impersonal form using various expressions, for example, various number systems like Roman, Arabic, decimal, binary, etc. Various transformations such as number-theoretic or spectral ones can be used here.

The next level of IT is data technology that allows us to collect information about objects creating statistical characteristics in the form of tables, summary information, etc. In other words, it is technology class that provides the ability to create databases.

The highest level of IT is represented by knowledge technology. They allow us to obtain knowledge and to create knowledge bases and knowledge-oriented systems. The comparative structure of the last two technologies [1] is shown in Fig. 2 that shows both similarities and differences between them. The interaction of these technologies in general will be discussed below.

CLASSIFICATION OF COMPUTING SYSTEMS

Considering the structure of information technology presented above we can proceed to the classification of computing tools and systems taking into account their functional features (Fig. 3). This classification shows that computing systems can be divided into two large classes: a class of deterministic systems and a class of chaotic systems [2, 3] or, in other words, systems with functional architecture.

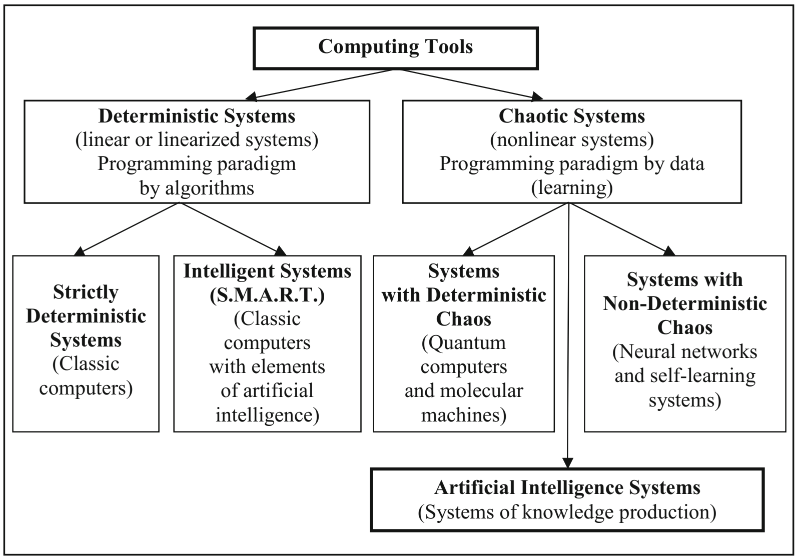

Fig. 3. Classification of computing systems based on IT.

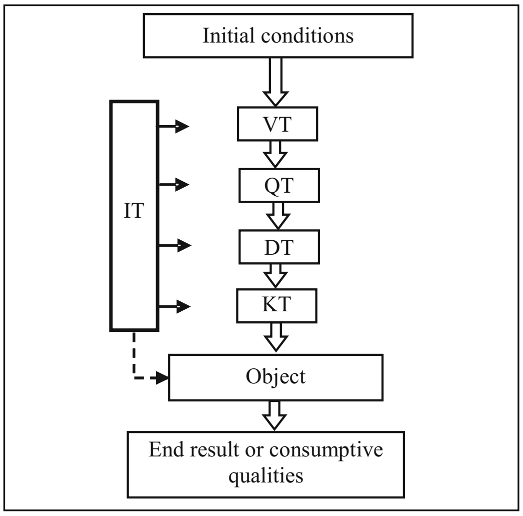

Fig. 4. Scheme of interaction of technologies in case of development of strictly determined systems (additionally shown for start systems by a dotted line).

Deterministic systems can be strictly deterministic systems and intelligent (S.M.A.R.T.) systems. The class of strictly deterministic systems includes conventional MFSMs, numerically controlled machines, computers with traditional architecture, etc. When developing deterministic systems both KT and DT are widely used. However, their use is limited by the fact that they are only a tool of the developer at the design stage. The scheme of using technologies to create strictly deterministic systems is shown in Fig. 4. Intelligent systems are the same strictly deterministic systems but with elements of artificial intelligence.

Intelligent systems (S.M.A.R.T.) [4] are designed in the same way but the system itself includes such IT components as recognition of situations, images, language commands, etc. In other words, these ITs become an integral part of the systems themselves.

The design theory of such systems is based on the principles of computer algebra [5] according to the following formula.

1. Computer algebra

$$U_M = \left\langle \bigcup_{i=0}^{N} u_i \right\rangle, \tag{1}$$

where u_i are algebraic systems, for which it is possible to build physically realizable models, and U_M is a computer algebra.

2. The command system is any signature of a computer algebraic system U_M

$$\Omega_M = < \mathbf{R}_M, \mathbf{F}_M, \mu_M >, \tag{2}$$

where $\mathbf{R}_M$ and $\mathbf{F}_M$ are the sets of symbols of machine relations and operations with which we can build a theory T_M (in other words, a method of organization of the computational process), μ_M is their virtual image.

3. Abstract architecture is an image of a computer algebra system on a physical plane that can model system operations and a method of organization of the computational process

$$T_M = f^{-1}(A, u_M) \text{ and } A = f(u_M, T_M), \tag{3}$$

where f is an interpretation function T_M in A, f^{-1} is an inverse interpretation function A in T_M.

According to what has been described above, it is possible to build an algebraic classification of deterministic systems presented in Table 1, respectively.

TABLE 1. Algebraic Classification of Computing Tools (R is a Predicate Formula, F is a Functional Formula, $\Rightarrow$ is a Semantic Implication)

Type of Computing Tool	**Form of a Chain of Formulas**	**Algebra of a Chain of Formulas**	**Algebra of a Signature of a Chain of Formulas**
Von Neumann machine	$F \Rightarrow$, $R \Rightarrow$	Algebra and model are separate commands. Algebra of functional formula: an additive group, a multiplicative group, and a semigroup	An additive group and a semigroup
Modern PCs	$F \Rightarrow$, $R \Rightarrow$	Algebra and model are separate commands. Algebra of functional formula: ring elements, an additive group, a multiplicative group, and a semigroup	An additive group and a semigroup
Signal processors 1 – of the 4th generation (Harvard architecture)	$F \Rightarrow$, $R \Rightarrow$	Algebra and model are separate commands. Algebra of functional formula: ring elements, an additive group, a multiplicative group, and a semigroup	An additive group and a semigroup
Signal processors of the 5th generation (Super Harvard architecture)	$F \Rightarrow$ $R \Rightarrow$ $R, F \Rightarrow$	An algebraic system. Algebra of functional formula: ring elements, an additive group, a multiplicative group, and a semigroup	An additive group and a semigroup
Signal processors (Super Neumann architecture)	$F, R \Rightarrow$	An algebraic system. Algebra of functional formula: a ring, a field, an additive group, a multiplicative group, and a semigroup	An algebraic system. A ring, an additive and a multiplicative group or a semigroup and a predicate

The second large class of tools of computing technologies is a chaotic system or systems with functional architecture. These systems are divided themselves into systems with deterministic and non-deterministic chaos.

Systems with deterministic chaos are based on the fact that some physical processes with given initial conditions either way converge with a high probability to a certain stable result, but usually such processes cannot apply the established data or knowledge technology due to the lack of possibility to reliably describe this physical process in any way. However, mankind has learned to use these processes without precise knowledge of their physical nature. But this is only a visible aspect of the problem. In fact, the main problem is that these computing processes are nonlinear. The traditional approach to solving nonlinear problems involves their linearization after which an algorithm for solving these problems can be developed. Because for whatever reason linearization is not always possible, in other words, data and knowledge technology are unattainable, «chaos» is used as a way to describe and solve these nonlinear problems. Hence, the name "chaotic systems". The scheme of interaction of technologies in this case is shown in Fig. 5. Here we can see that knowledge and data technologies reside in the most nonlinear object that is used to solve such problems.

Note that in practice this method of solving nonlinear problems is not accepted as it is difficult and sometimes impossible to find objects with the desired features. It is easier to replace nonlinear objects with mathematical models. Neural networks or perceptrones or ordinary computers with programs that use chaotic algorithms are most often used as models.

Let us proceed to the concept of systems with non-deterministic chaos. The scheme of use of technologies in this case is presented in Fig. 5, where the nonlinear object is replaced by its mathematical model. Note that the use of neural networks in the development of foundations of computing technologies at the present stage is very successful. However, this success is not as much due to their use in solving artificial intelligence problems but more to the fact that they allow us to solve nonlinear problems. This raises the question of how to specify a nonlinear object if its analytical expression is unknown. The answer is given by nature as this method is learning, i.e., selection of parameters through trial and error. For technical systems, we obtain a paradigm shift in programming: we pass from programming by algorithms to programming by data.

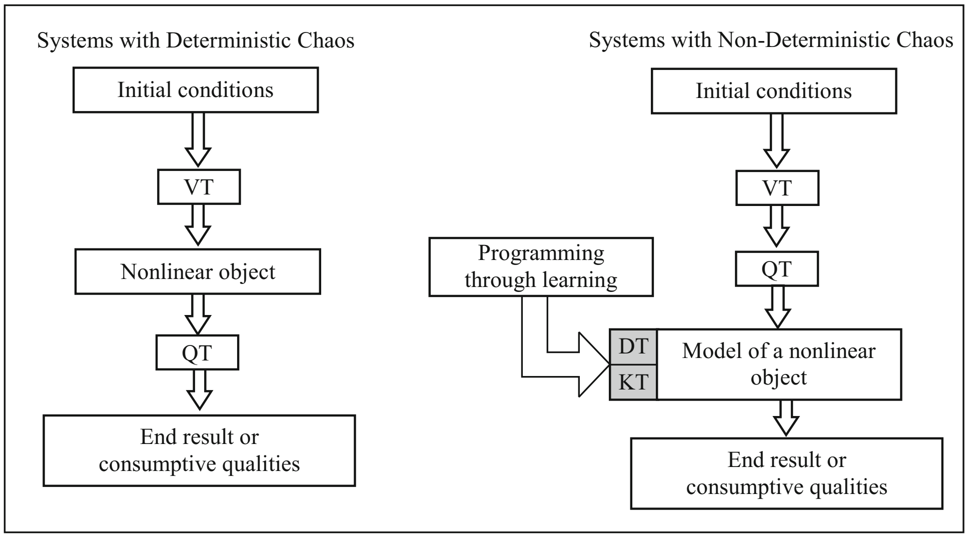
Systems with Deterministic Chaos
Initial conditions
VT
Nonlinear object
QT
End result or consumptive qualities
Systems with Non-Deterministic Chaos
Initial conditions
VT
QT
Programming through learning
DT
KT
Model of a nonlinear object
End result or consumptive qualities

Fig. 5. Scheme of interaction of technologies in case of development of chaotic systems.

PROGRAMMING PARADIGM AND SELF-LEARNING DIGITAL AUTOMATA

At first we need to specify the concept of the algorithm of problem solving. Any reduction of any formula or analytical dependence becomes an algorithm when it is written in the form of binary operations and binary relations.

Let us consider some features of genetic algorithms widely used at the present stage. These features were obtained as a result of a number of investigations that were previously not sufficiently considered. They are the following:

- use of multi-invariant expression of the same data (for example, a sign of division into any number);
- invariance in relation to any shift or permutation, use of combinatorics (click reaction $2^n - 1$ [6]);
- invariance in relations to both functional transformations and predicate dependence (for example, division in arithmetic);
- reduction of special dimension of expressions (use of symmetry, holograms, and fractals);
- virtualization of expressions (one image equals one word);
- change of programming paradigm (programming by data);
- a phenomenon of replication as a method of developing high integrity systems.

Analysis of these features postulates shows that although the number of data processing algorithms present in a neuron DNA is big, it is still restricted i.e., finite. If a shortage of DNA algorithms occurs, then it will be compensated by multi-invariant expressions of the same data. Hence, the computing power of every algorithm can be computed as

$$N = \sum_{i}^{l} r_i \cdot k, \tag{4}$$

where r_i is a number of reductions of this algorithm, k is a number of possible invariant data representations for this algorithm. In practice, the value N can be unrestricted, i.e., the search for an algorithm can become unrealized in a given time due to a big number of variants. Hence, nature or evolution chose a different path of changing the programming paradigm from the performance by programming algorithms to the performance by data algorithms, i.e., learning. One of the types of programming by data is based on building tables. If the table is built arbitrarily (i.e., is disordered, for example, when obtaining experimental data) for a large quantity of data, then it is impossible to find an algorithm of building of this table. Then the continued use of the table can be considered a process of learning. This principle of learning, or rather pseudo-learning can be used when developing simple self-learning MFSMs. In Fig. 6, a combinational logic circuit able to perform programming by data, i.e., to learn as it was not created by an analytical method or with help of a truth table but by a method of feeding it information in an

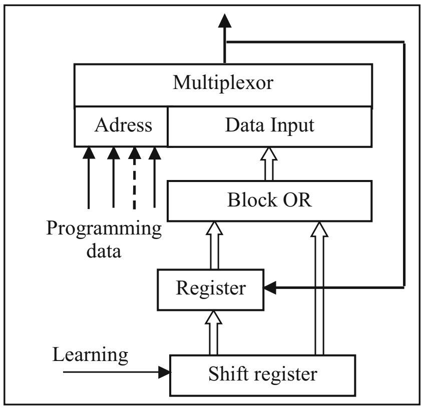

Fig. 6. Self-learning logic element.

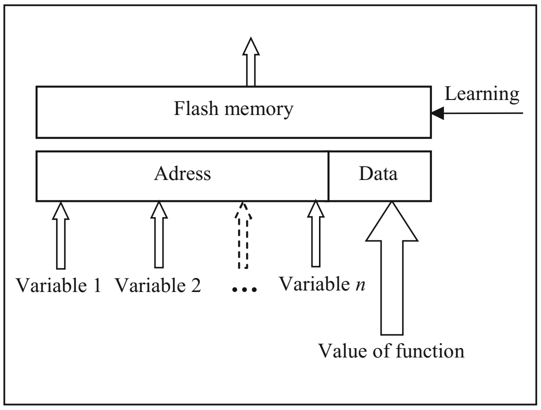

Fig. 7. Functional self-learning elements.

arbitrary order. This circuit functions in the following way. The base of the element is found on a multiplexor with two registers one of which is a shift register and the other one is a set of triggers learning the value of a logic function. In order for this circuit to learn a code that that has to be learned by the circuit is fed to the address input of the multiplexor. Then the shift register with digit one running through it is started. In a case when the address of the multiplexor coincides with its data input digit one also appears at the output of the multiplexor and is memorized by a corresponding trigger value of the logic function. For a large body of data the process of memorization coincides with learning because there is no need to search for the algorithm that created this logic function. Evidently, for the learning process of any function except for logic ones an assembly of logic elements can be used that is then going to be transformed into flash memory or a programmable logic device (PLD) (Fig. 7). Hence, the analysis performed in this section and a modular model of neurons [7] allow us to build corresponding neural networks based on already investigated MFSM and circuits of computing technologies and change the programming paradigm from programming by the algorithm to programming by data.

COMPUTING SYSTEMS WITH CHAOTIC ARCHITECTURE

Let us consider computing organization in chaotic systems. It is obvious [2] that these systems are able to develop themselves and need to have a certain free computing resource serving as a corresponding motivation. However, a restriction placed onto this free resource is a certain command system. By its nature the system restricts means of development as it is cannot expand (the field of the operation code has a fixed length measured in bits). Hence, expansion of computing capabilities is provided by the method of virtualization at the expense of reduction of productivity. To solve this problem a model of Post-Turing machine can be used with the necessary interpretation. Assume computing machine consists out of a body functional and logic elements in the form of a PLD, an interface between its memory and the PLD, the memory itself, a compiler for the PLD with a command interpreter and its database, a control processor, and a learning node (Fig. 8). In the control processor, only transfer instructions are hardware-realized like, for example, in a Post-Turing machine (only transfer instructions between the memory and the PLD are considered here). With the help of the compiler of the PLD a user of this machine can create any unit, for example, an arithmetic block or a computing function, etc. Then he places it into the address space of the machine in a way that the data inputs of these units have their separate address in the address space. After that by transferring data from the memory into the created units a number of operations can be performed. Created operations are placed into the database of the interpreter. The interpreter will then interpret the program written in a high level language in order of transfer. Hence, the program will consist of transfers only. As a user creates a command system as they wish this command system, as well as the architecture, will be chaotic because it will differ from a command system created by another user using the same system. Thus, we can create computers with minimized sensitivity towards viruses, i.e., we will be able to provide cybersecurity for our programs.

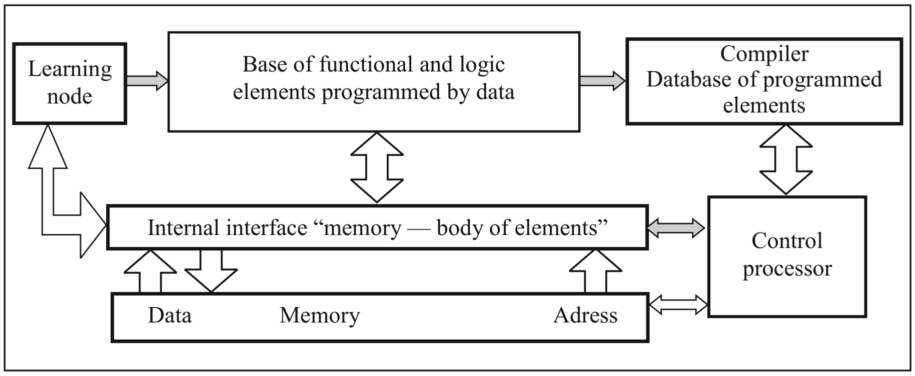
Learning node
Base of functional and logic elements programmed by data
Compiler Database of programmed elements
Internal interface "memory — body of elements"
Control processor
Data
Memory
Adress

Fig. 8. Processor with chaotic (functional) architecture.

A question arises: what to do in a case when a user does not have enough knowledge in the field of creating computing machines? The answer lies in including a neural network into the system. A user can teach this network to solve corresponding problems using the method of learning. However, after the learning process the neural network becomes a deterministic system. Hence, by developing a modular model of neurons we can technically create an algorithm of solving a number of problems. Then having obtained the algorithm, we can synthesize the needed computing units with the help of a PLD compiler. Since these transformations are formal, the process of algorithm creation can be automated. In favor of this approach, the user will not know their command system, but changing the computing units as needed the user will develop the system according to their needs.

INTERACTION OF IT LEVELS AND ARTIFICIAL INTELLIGENCE SYSTEMS

Let us consider the interaction of levels of information technology with a simple example. Assume we need to calculate a certain body weight. To do this we will use Newton's law $F = ma$. This law is undoubtedly a component of knowledge technology. However, based on this law, the body weight is determined by a formula $P = mg$. To calculate the weight we need to know the body mass m and the acceleration of gravity g. It is known that the acceleration g is not a constant value for all points on the Earth's surface, so you need to obtain its value for the selected point. In addition, you should choose a system of units for both acceleration and body mass. Thus, we have to go down to the level of data technology. Next we need to perform calculations, which means that we need to choose a number system in that the calculations will be performed, i.e., the presentation of data. With that, we need to pass to another level below, i.e., to the level of quantitative technology. After calculating the body weight we need to interpret the obtained results, i.e., move lower to another level (the level of virtualization technology), because virtualization is one of the types of interpretation of facts.

Note that the process of knowledge formation [8] can also be performed in the same way. First a hypothesis needs to be formed on the basis of axiomatics as a coincidence of facts taking place. Then we need to prove that this coincidence of facts is not accidental and takes place under any circumstances. When such proof is established it means that the hypothesis has passed into production where cause and effect are connected by a semantic sequence that is not subject to the law of contraposition. In other words, if the cause is true then the effect is true as well. Currently this approach is often used to create quantum computers, molecular machines, etc. However, this does not mean that the inverse statement is true as well (when the effect is true then the cause is true as well). It needs to be proved and then the semantic sequence will pass into a logical sequence. In the language of mathematics this means that the theorem has been proven. Next, to develop a theory we need to learn all the reductions of the obtained dependence and to generalize them. This is how knowledge is created. This example shows that we manipulate certain levels of information technology going lower and lower each time.

However, there exists another way, which is the movement from the lower levels of technologies to the higher ones. Let us consider it as well. To do this, we will clarify the concept of systems of artificial intelligence. Assume that a system has artificial intelligence when it is able to produce a technology of knowledge. Obviously, the way of knowledge acquisition through hypotheses is not accepted in this case, so we will use the reverse way, i.e., we will move

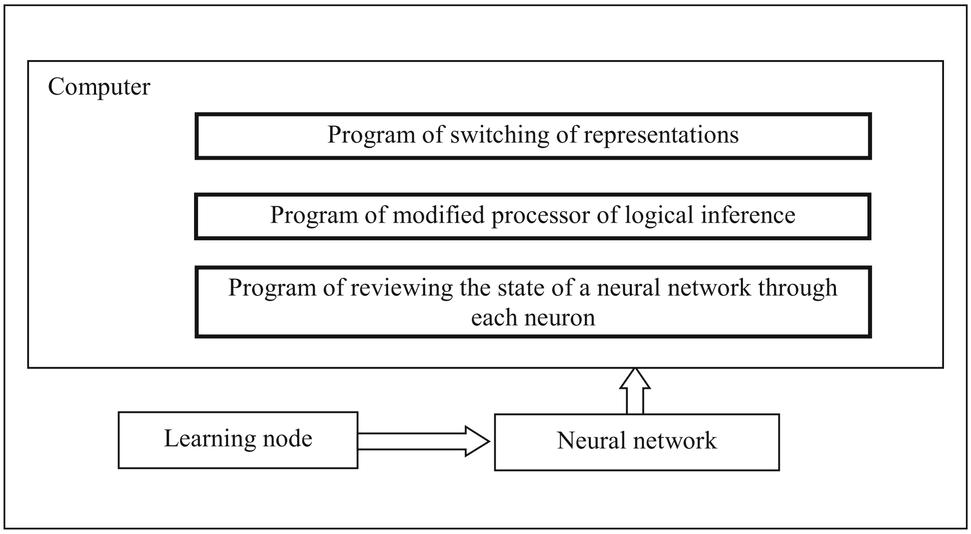
Computer
Program of switching of representations
Program of modified processor of logical inference
Program of reviewing the state of a neural network through each neuron
Learning node
Neural network

Fig. 9. The scheme of knowledge acquisition.

on the levels of technologies from the bottom to the top. To do this, we will use a neural network and teach it how to solve a simple problem. After the learning process the network will become a deterministic device, as it will determine all the necessary thresholds of neurons, as well as the transfer coefficients and connections between the neurons of the neural network. If we analyze these parameters with the help of quantitative technologies we will be able to find an algorithm for solving this problem.

As a method satisfying the requirements of knowledge acquisition at the first stage, the scheme of reviewing each neuron in the neural network and describing each neuron using one or more production rules is used. Obviously, the type of obtained production rules depends primarily on the type of nonlinear function of the neuron and signal enhancement or signal attenuation (continuous or discrete) passing through the neuron. These production rules are then subjected to statistical analysis using programs that form a modified logic output processor or a neuroimitator that are used to verify a production in order for the semantic sequence to become a logical sequence. If this is achieved, then the first stage is complete. Then these results are brought together and presented in a matrix form to users for an investigation.

The second stage that completes the process of knowledge acquisition is an extraction of representations of specific parameters (quantitative features) from the obtained dependencies in such a way that they are independent of these representations. The scheme illustrating this process as a whole is given in Fig. 9.

CONCLUSIONS

The IT analysis shows that these technologies generally have the following four levels of IT: virtualization technology, quantitative technology, data technology, and knowledge technology. Hence, these levels divide the computing tools into two classes with a significant difference: a class with traditional architecture and a class with chaotic architecture. The main aspect of the distribution is a paradigm shift in programming, i.e., the replacement of programming by algorithms with programming by data (learning). As the learning process is not strictly deterministic and always takes place in different conditions, the architecture of these computing technologies becomes chaotic. In other words, a new direction in the development of computing technologies is formed, namely chaotic architectures, where not only the architecture of the tools of the technologies itself is chaotic, but also their commands systems, allowing us to create a new class of the information systems that are protected from cyber-attacks and have other useful features.

REFERENCES

1. S. H. Park and M.W. Suh, "Data technology and e-statistics with their applications in industry," in: Proc. 4th Asian Regional Section of the International Association of Statistical Computing (Busan, Korea, December 5–7, 2002), Busan (2002), pp. 201–204.
2. Yu. V. Andreyev, A. S. Dmitriev, and D. A. Kuminov, "Chaotic processors," Advances in Modern Radioelectronics (Foreign Radioelectronics), No. 10, 50–79 (1997).
3. G. T. Doran, "There's a S.M.A.R.T. way to write management's goals and objectives," Management Review (AMA FORUM), Vol. 70, Iss. 11, 35–36 (1981).
4. W. Ditto, S. Sinha, and K. Murali, "Method and apparatus for a chaotic computing module," US Patent No. 07096347, August 22 (2006).
5. M. V. Semotiuk, Notes on Machine Algebra [in Russian], Stal', Kyiv (2012).
6. J. Z. Tsien, "A postulate on the brain's basic wiring logic," Trends Neurosci., Vol. 38, Iss. 11, 669–671 (2015). https://doi.org/10.1016/j.tins.2015.09.002.
7. V. I. Klyukin and Yu. K. Nikolaenkov, Neural Network Structures and Technologies. Part 1: Electrical and Mathematical Models of Neurons. Feedforward Neural Network [in Russian], Voronezh State University Publ. House, Voronezh (2008).
8. V. G. Lukianets, "Automation of the process of knowledge extraction," in: Engineering Education: Challenges and Developments: Proc. VIII Intern. Sci. and Method. Conf. (Minsk, November 17–18, 2016), Part 2, BSUIR, Minsk (2016), pp. 11–13.