Вестник РУДН. Серия: Инженерные исследования
RUDN Journal of Engineering Research

2023;24(4):349–364

ISSN 2312-8143 (Print); ISSN 2312-8151 (Online)
journals.rudn.ru/engineering-researches

DOI: 10.22363/2312-8143-2023-24-4-349-364
UDC 519.1:519.681:519.683.8:519.685.1
EDN: HBEUFG

Research article / Научная статья

The synthesis of structural diagrams of automatic devices on formal neurons

Natalia L. Malinina

Moscow Aviation Institute (National Research University), *Moscow, Russian Federation*
malinina806@gmail.com

Article history
Received: April 24, 2023
Revised: August 20, 2023
Accepted: September 22, 2023

Conflicts of interest
The author declares that there is no conflict of interest.

Abstract. The development of finite state machines and the synthesis of neural networks come with enormous computational difficulties. The problems that are faced both by the creators of control finite state machines and the creators of neural networks are almost the same. In order for a control finite state machine to be implemented, an algorithm for its operation must be created, and then a program must be written, and finally this program must be implemented in hardware in the form of a finite state machine. It is crucial to create a finite state machine, which will be deterministic. As for neural networks, it is necessary either to set the weights on its edges with the help of experts, or it must be trained to obtain optimal weights on its edges. Both tasks, that is, the determination of finite state machines and the training of neural networks, are currently most often performed using approximate (exponential or genetic) algorithms. At the same time, few authors point out the fact that, firstly these algorithms give an error of up to 15 %, and secondly the operating time is quite long and requires large energy costs. The article has proven that control finite state machines and neural networks are equivalent based on their structure, which can be represented as a directed edge graph. Such equivalence makes it possible to use methods of normalizing arbitrary graphs to determine finite automata and synthesize neural networks. Methods of graph normalizing are extremely new, they are based on a fundamentally new approach of the extension of graph theory and will allow performing these operations using algorithms that have linear complexity or can significantly reduce the number of options when using brute force.

Keywords: finite machine, determination, neural network, directed graph, normal algorithm

For citation
Malinina NL. The synthesis of structural diagrams of automatic devices on formal neurons. *RUDN Journal of Engineering Research.* 2023;24(4):349–364. http://doi.org/10.22363/2312-8143-2023-24-4-349-364

Синтез структурных схем автоматических устройств на формальных нейронах

Н.Л. Малинина

Московский авиационный институт (национальный исследовательский университет),
Москва, Российская Федерация
malinina806@gmail.com

История статьи

Поступила в редакцию: 24 апреля 2023 г.
Доработана: 20 августа 2023 г.
Принята к публикации: 22 сентября 2023 г.

Заявление о конфликте интересов

Автор заявляет об отсутствии конфликта интересов.

Аннотация. Разработку конечных автоматов и синтез нейросетей сопровождают огромные вычислительные трудности. Проблемы, с которыми сталкиваются как создатели управляющих конечных автоматов, так и создатели нейросетей, практически одинаковы. Для того чтобы управляющий конечный автомат мог быть реализован, надо сначала создать алгоритм его работы, потом написать программу, потом эту программу реализовать в «железе» в виде конечного автомата. Главное — надо создать, и это важно, детерминированный конечный автомат. Что касается нейросетей, то, чтобы она работала, необходимо либо задать с помощью экспертов веса на ее ребрах, либо ее надо обучить, чтобы получить оптимальные веса на ребрах. И то, и другое, то есть, детерминизация конечных автоматов и обучение нейронных сетей, в настоящее время производится чаще всего с помощью приближенных (экспоненциальных или генетических) алгоритмов. При этом часто авторы не указывают на тот факт, что, во-первых, эти алгоритмы дают ошибку до 15 %, а, во-вторых, время работы подобных алгоритмов достаточно велико, и требует больших энергетических затрат. В материале статьи доказывается, что управляющие конечные автоматы и нейросети — эквивалентны, если исходить из их структуры, которую можно представить в виде ориентированного реберного графа. Подобная эквивалентность позволяет применять для детерминизации конечных автоматов и синтеза нейросетей методы нормализации произвольных графов. Методы нормализации произвольных графов новые, они основаны на расширении теории графов и позволят применять алгоритмы линейной сложности или существенно уменьшать число вариантов при переборе.

Ключевые слова: конечный автомат, детерминизация, нейросеть, ориентированный граф, нормальный алгоритм

Для цитирования

Малинина Н.Л. Синтез структурных схем автоматических устройств на формальных нейронах // Вестник Российского университета дружбы народов. Серия: Инженерные исследования. 2023. Т. 24. № 4. С. 349–364. http://doi.org/10.22363/2312-8143-2023-24-4-349-364

Introduction

A finite state machine (FSM) is an extremely simplified model of a computer, having a finite number of states and sacrificing all the features of computers, such as RAM, read-only memory, input-output devices and processor cores in exchange for ease of understanding, ease of reasoning and ease of software or hardware implementation. It can be said that FSM is an algorithmic component of a "data-less" program that models "instinctive" behaviour that is not adaptable to the sequence of environmental influences. In other words, FSM are technologies designed to facilitate the development of other algorithms; they serve as a means of achieving the ultimate goal — the implementation of the algorithm. A neural network is a computational or logical circuit built from homogeneous processing elements, which are simplified functional models of neurons. The transfer functions of all neurons in a

neural network are fixed, and the weights are parameters of the neural network and can be changed. Neural networks are trained using genetic or other exponential algorithms.

Such algorithms take a long time to work, take up a lot of memory, and, moreover, are not absolutely accurate. Any neural network is a finite state machine and any finite state machine can be replaced by a suitable neural network [1]. The equivalence of the structures of finite state machines and neural networks makes it possible to solve the problems of their structural synthesis using the same methods. The problem of structural synthesis, both of a DFSM and a neural network, belongs to the area of NP — hard.

1. Finite State Machines: the basic concepts and problems

Finite state machine (FSM) is a model of a computing device with a fixed and finite amount of memory. They read and process a chain of input symbols belonging to a finite set. Among the first researchers in the search of the simplest models of finite state machines were McCulloch and Walter Pitts, who proposed a concept similar to a finite state machine in 1943 [2].

An autonomous FSM, starting from a certain chart (diagram), can only generate a periodic sequence of x states. Such sequential execution of a given cycle of operations is typical for many areas of modern technology, therefore, the dynamic systems, which in an acceptable idealization can be considered as an autonomous FSM, are widely applied particularly for the implementation of an automaton approach to programming. The theory of formal languages [3–7] may be used for their design. And, finally, and most importantly, autonomous FSM are used in the synthesis of logic control algorithms [8-12].

The finite state machine transforms the input character sequences into the state or the output character sequences. Theoretically the deterministic state machine (DFSM) can becreated from non-deterministic state machine (NFSM) according to reduction of DFSM to NFSM (Kleene's theorem [13]). Since the number of states (output symbols) is finite, the question is: what input sequences cause each of the possible states (or each of the output symbols) to occur? The answer was given by Kleene's theorems [13], which established that only the events of the regular sets can be represented in a finite state machine. In this case, an algorithm for constructing any regular sets can be established.

However, in practice, determination is not always possible, since in the worst case the number of states in an equivalent DFSM grows exponentially with the increase in the number of states of the original NFSM. This situation becomes the main problem when creating algorithms for reducing the NFSM to the form of a DFSM.

So, the main problem arises: how to make a set regular? The set becomes regular if it can be ordered. And there is no efficient way to understand whether the set is regular or not. Limitations on the capabilities of computers (Gödel's theorem [14]) made it necessary to use technologies of genetic (evolutionary) or other exponential algorithms in order to create DFSM.

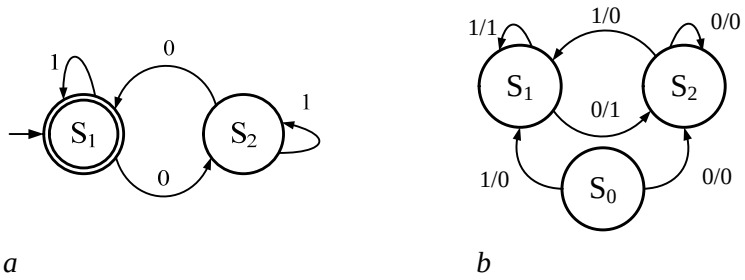
Thus, the finite state machines are classified as the deterministic (DFSM) and the non-deterministic (NFSM). The only and main difference between NFSM (non-regular set) and DFSM (regular set) is the existence of several transitions in one symbol from one state. A deterministic finite automaton is one in which, for any given sequence of input symbols, there is only one state to which the automaton can go from the current state.

If the class of the dynamical systems can be extended in order to include infinite memory, then for the dynamical systems of this wider class (Turing machines) the answer to the question "what can they do?" is much simpler — they can implement any predefined algorithm. The concept of a Turing machine underlies the definition of the concept of an algorithm: an algorithm is any process that can be carried out on a finite state machine supplemented with the infinite memory, that is, algorithmically complete machines: on a Turing machine [15], on a Post machine [16], etc.

According to the above definition, deterministic finite automata are always complete — they define a transition for each state and for each input symbol. In addition, to ensure the uniqueness of the algorithms, the synthesised finite automata must be deterministic (ordered). When developing programs that are characterised by complicated control logic, one can use the automaton approach, which allows making the program text more regular and compact.

Finite state machines (FSM) can be represented as block diagrams; it is the traditional technology for algorithms. The most convenient form of their

representation for a person is a graphical one — a state-transition-diagram, and for programming and formal transformations — a tabular one. State diagrams provide a graphical way to model how a system responds to a disturbance, and is a graph. It specifies how the system can move from one state (vertex) to another one. A key characteristic of such event-driven systems is that the behaviour of the system often depends not only on the last or current event, but also on the previous events, which is expressed using a state diagram. So, a state diagram for a finite state machine is an ordered graph in which the vertices denote states, and the arcs show transitions between two states (Figure 1).



a *b*

Figure 1. The graphical representation of the finite state machines: a — S_1, S_2 — states. The arces are labeled by input data; b — S_0, S_1, S_2 — states. The arces are labeled as j/k, where j — input data, k — output data
S o u r c e : made by the author

In terms of graph theory, the problem of covering all transitions of the automaton is formulated as the task of graph traversal, that is, passing along a route containing all the arcs of the graph. There are two main problems associated with the graph traversal for the automaton state machine: nondeterminism and too large size of the graph. A nondeterministic automaton is an automaton in which the transition function is ambiguous: one pair corresponds to several arcs in the graph. Since the choice of one or another of these arcs cannot be determined by the test action, it is impossible to guarantee the unambiguous traversal of the state graph during testing. Although it should be noted that the ambiguity of the exit function does not create additional problems. It is only required that some predicate from the state, input, and output symbols be satisfied. [12]. The evaluation of the DFSM generation algorithm is disappointing. The process of generating a DFSM is a NP — hard problem. But on the other hand, after the DFSM is generated, it processes any symbol in constant time and a string of length N in $O(N)$ time.

An NFSM in a state-transition diagram can have two or more arcs as outputs from the same state labeled with the same input symbol. Such a NFSM does not have an adequate tabular (functional) representation, but can be transformed into a conventional DFSM. The lack of an internal memory limits its ability to transform chains (simulation ability). Although in the general case such restrictions allow to solve many problems.

The are also controlling FSM. Their main difference from other types of FSMs (transformers and recognizers) is that they contain not separate input actions, but Boolean formulas from them [18–21] in the transition marks. However, the construction of control FSM is even more difficult, and in some cases, it is not possible to build such an automata at all.

The ordering of control FSMs using the exhaustive enumeration, even with small sizes of FSM, is extremely time-consuming, and their heuristic construction does not always give acceptable results, although sometimes this is the only way. The simulated annealing method does not provide a significant improvement [20]; ant algorithms are more suitable for problems where solution is to find paths in a graph. Therefore the most of the work in the field of software search engineering is based on the use of evolutionary algorithms [21]. One of the most important adventages of genetic (evolutionary) algorithms is the absence of the need for information about the behaviour of the function and the negligible impact of possible gaps on optimisation processes.

Genetic algorithms are also used to increase the efficiency of neural networks training. The explo-

sive growth of interest in artificial intelligence (AI) is mainly due to the growth of computing capabilities with its help, and not to the emergence of new algorithms.

The task of structural synthesis is to construct an automaton diagram of minimal complexity. It should be noted that the problem itself is *NP* — hard. Gödel's theorems [14] are directly related to the limitation of the capabilities of computers, which also were recognized by Turing, Church, Nagel and others. Turing showed that the same restrictions apply to humans [15]. In this regard, neural network technologies began to be used to create FSM, that is, researchers began to create FSM, using neural networks or genetic (evolutionary) algorithms.

2. Neural network. The basic concepts. Problems

Recently, more and more people began to discuss neural networks, and great attention is being paid to the creation of artificial intelligence based on artificial neural networks. A lot of attention is being paid to this scientific area. Let us briefly review the principles that are embedded in automatic neural networks.

A biological neuron has processes of nerve fibers of two types: dendrites, through which impulses are received, and an axon (it is the only one), along which a neuron can transmit an impulse. The axon contacts the dendrites of other neurons through special formations — synapses, which affect the strength of the impulse. It can be assumed that during the passage of the synapse, the strength of the impulse changes a certain number of times, which is called the weight of the synapse. Impulses received by the neuron simultaneously through several dendrites are summed up. If the total impulse exceeds a certain threshold, the neuron is excited, generates its own impulse and transmits it further along the axon. It is important to note that the weights of synapses can change over time, which means that the behaviour of the corresponding neuron also changes.

Neural networks are artificial, multilayer, highly parallel logical structures made up of formal neurons. The foundation of the theory of neural networks and neurocomputers was laid by the work of American neurophysiologists [2]. The book [22] had a significant influence on the further development of the neural network theory. The theory of neural networks continues to develop quite intensively at the beginning of the 21st century. Potential areas of application for artificial neural networks are those where human intelligence is inefficient and traditional computations are time-consuming or physically inadequate. The relevance of the use of neural networks increases many times when it becomes necessary to solve poorly formalized problems. The main areas of application of neural networks: automation of the classification process, forecasting, recognition process, decision-making process; management, coding and decoding of information; approximation of dependencies, etc. With the help of neural networks, an important task in the field of telecommunications is successfully solved — the design and optimisation of communication networks, as well as the tasks of designing new telecommunication networks.

Thus, the creation of automatic systems based on a neural network consists of choosing the network architecture and the selecting of the network weights. The selection of weights is the process of "training" the network. Neural networks turn out to be something between a central processing unit and a human brain. At this moment the selection of weights is carried out using either genetic algorithms or expert assessments. So, a neural network is a computational or logical chart built from homogeneous processor elements, which are the simplified functional models of neurons (Figure 2).

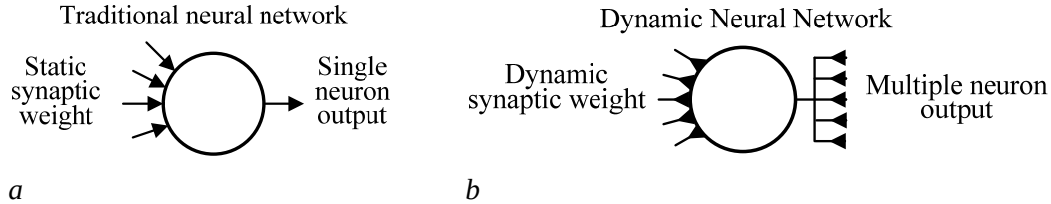


Figure 2. The graphical representations of the neural networks:
a — Traditional neural network; *b* — Dynamic neural network
S o u r c e : made by the author

As a rule, the transfer functions of all neurons in a neural network are fixed, and the weights are the parameters of the neural network and can change. Some inputs of neurons are labeled as external inputs of the neural network, and some outputs are labeled as external outputs.

The work of the neural network is to transform the input signal into an output signal, and this transformation is determined by the weights of the neural network. A formal neuron in neural networks is a processor element, which is a data converter that receives input data and transforms it in accordance with a given function and parameters. A synapse in neural networks is a connection between formal neurons. The output signal from a neuron enters the synapse, which transmits it to another neuron. Complicated synapses can have memory. As a rule, there are quite a lot of synapses in a neural network. An adder in neural networks is a block that sums up the signals coming from neurons through synapses. In a general case, an adder can transform signals and transmit them to neurons or adders also through synapses.

3. Graphical representation of the main elements of a finite machine on formal neurons

Let the functions of the finite automaton be given. It is necessary to build its block diagram on formal neurons. Thus, it is necessary to show the connection between the structure of an ordinary network model and the structure of an automaton based on formal neurons (AFN). To solve this problem, an ordinary normal network models can be used [23], since they contain the vertices of only well-defined types. Structural-optimal network models are best suited since they do not have the simplest vertices in which no logical functions are implemented. It is known that the canonical edge graph has four types of vertices [23]. These four types of vertices are similar to the main types of structural formations of the nervous system of living beings, which includes:

- Receptors — nerve endings that transmit external excitations to the nervous system (cells that have only outputs);
- Neurons — are nerve cells with $m_n \geq 1$ inputs (dendrites) and only one output (axon) (a formal neuron has the same structure [23];
- Branching of axons that transmit nerve excitations to other neurons and tissues, that is, the elements of the nervous system that have one input and $m_a \geq 1$ outputs;
- Effectors (endings) that transmit nerve excitations to the working organs, that is, cells that have only inputs from the nervous system.

The obvious analogy between the structure of a canonical edge graph and that of a neural network, as well as between the structure of one of the types of vertices of a canonical edge graph and that of a formal neuron, leads to the reasonable assumption that canonical edge graphs can be used in order to:

- Formalise the synthesis of the structure of automatic devices and systems built on formal neurons;
- Formalise the synthesis of mathematical models that would enable studying individual functions of the nervous system of a living organism, implemented in accordance with a given logic and a set of external excitations.

The stated assumption is quite consistent with the theorem given in [24] that any finite automaton can be replaced by a suitable network of formal neurons.

The question arises: is it possible, using the principle of normalizing, to transform the diagram of a given finite machine into a network consisting of formal neurons? Using the example of normal algorithm synthesis [23], it was shown that operators can be represented as arcs of a canonical edge graph, connecting vertices of certain four types. Thus, it turns out that there is a similarity between the structure of the nervous system and the structure of normal algorithms. This similarity allows us to suggest that the basis of nervous activity, including higher, apparently, is a process similar to a normal algorithm, although, most likely, everything happens the other way around: normal algorithms intuitively reflect the internal activity of the nervous system.

Another consideration leads to these assumptions. Any change in the canonical system of binary relations by introducing additional links (pairs (q_i, q_j) of elements) violates the canonicity of the system of binary relations and requires its normalisation, which is associated with an increase in the order of the matrix and a change in the structure of the graph and the corresponding normal algorithm. Probably something similar occurs in the process of higher nervous activity. A new external stimulus or new needs, which the body's response to these stimuli must meet, is equivalent to establishing new connections in the brain. This violates the “normal

algorithm" existing in the brain and requires its new normalisation, which is because new brain cells are involved. All this is similar to the Δn — transformation of the adjacency matrix until the "algorithm" becomes normal again. One more suggestion should be added. The structure of a normalized matrix always depends on what new links need to be normalised. The emergence of the required new connections will determine the nature and the result of the work of the new algorithm of the nervous system model.

4. The formulation of the problem

From all that has been said above, it is clear that any neural network is a finite state machine. Likewise, any state machine can be replaced with a suitable neural network. Therefore, the problems of structural synthesis, both for the neural networks and the finite machine, can be solved by the same methods.

Consequently, one can try to build a model of the nervous system that organises itself under the influence of external stimuli and find expedient external stimuli and methods of self-organisation. The above assumptions may seem very bold, but automatic devices based on formal neurons have a very high reliability [24], so the solution of the problem of their synthesis is certainly very important and relevant.

Let us consider an example of synthesising a block diagram of a finite machine on formal neurons. In terms of network models, a formal neuron, together with its output (axon), can be represented as an arc of a directed graph, the initial vertex of which must be of the second type, and the final vertex must be of the third type. Let us give conventional names to the types of vertices of the canonical edge graph (Figure 3, *a*) by analogy with neural networks:

- Vertex of the first type is a receptor.
- Vertex of the second type is a neuron.
- Vertex of the third kind is an axon.
- Vertex of the fourth type is an effector.

Note that these vertices are considered together with their inputs and outputs. Each of these vertices can be connected by an arc with the other vertices (Figure 3).

Possible connections of neural network vertices are shown in the matrix in Figure 4, *a*. The same figure shows the second matrix (Figure 4, *b*), where the units are replaced by conventional signs that determine the nature of the connections of the arc connecting the nodes of the neural network with other arcs.

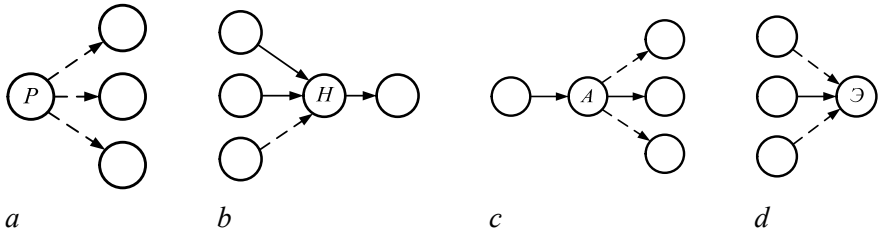


Figure 3. Types of the neural network vertices:
a — Receptor: $\rho^{(-)}(\text{р}) = 0$; $\rho^{(+)}(\text{р}) \geq 1$; *b* — Neuron: $\rho^{(-)}(\text{н}) > 1$; $\rho^{(+)}(\text{н}) = 1$;
c — Axon: $\rho^{(-)}(\text{а}) = 1$; $\rho^{(+)}(\text{а}) \geq 1$; *d* — Effector: $\rho^{(-)}(\text{э}) \geq 1$; $\rho^{(+)}(\text{э}) = 0$
S o u r c e : made by the author

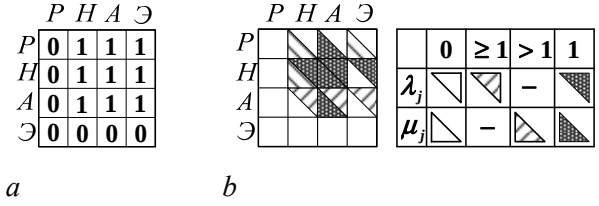


Figure 4. Possible neural network connections:
a — Matrix of the connections of neural network vertices;
b — Matrix with conventional signs that determine the nature of the connections of the arcs, and the table with conventional signs
S o u r c e : made by the author

Let us introduce the following characteristics of arc connections in a neural network:

- $\lambda_{j=i}$ — is the specific "load" of the j-th arc coming out of some vertex, from each of the arcs entering the initial vertex of the j-th arc;
- $\mu_{i=j}$ — is the "participation share" of the i-th arc in the "load" of all arcs emerging from the vertex, which is the end of the i-th arc.

The calculation of $\lambda_{j=i}$ and $\mu_{i=j}$ is performed as follows. For a normal adjacency matrix, the matrix $\|\omega_{ij}\|_1^n$ of the "relative weights" of the elements of the adjacency matrix is calculated.

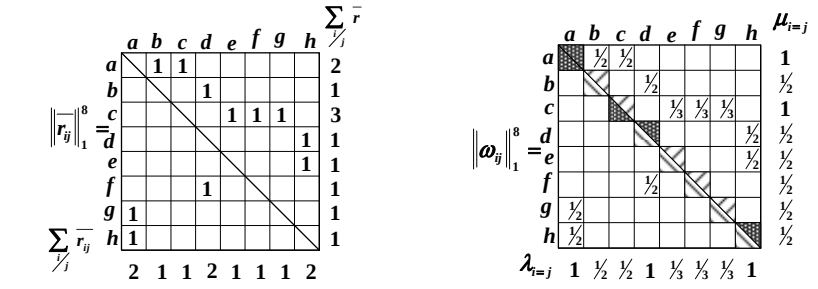
For each $r_{ij} = 1$ the corresponding value of ω_{ij} will be:

$$\omega_{ij} = min\left[\frac{1}{\sum_{i=1/j}^{n} r_{ij}}; \frac{1}{\sum_{j=1/i}^{n} r_{ij}}\right].$$

Then:

$$\lambda_{j=i} = \sum_{i=1/j}^{n} \omega_{ij}\,;\ \mu_{i=j} = \sum_{j=1/i}^{n} \omega_{ij}.$$

For example, let the matrix $\|r_{ij}\|_1^n$ be given (see Figure 5, *a*). For this matrix let us calculate the matrix $\|\omega_{ij}\|_1^n$ and the values $\lambda_{j=i}$ and $\mu_{i=j}$ (Figure 5b). Values $\lambda_{j=i}$; $\mu_{i=j}$, as well as $\sum_{i=1/j}^{n} \bar{r}_{ij}$ and $\sum_{j=1/i}^{n} \bar{r}_{ij}$ are calculated for each row of the matrix $\|r_{ij}\|_1^8$, completely determine the types of the initial and final vertices of any arc of the edge graph corresponding to one or another row of the matrix. Since an axon can have one or more outputs, the total number of options for arcs connecting the vertices of the neural network is 16. All these options are presented in a matrix and graphical form in Figure 6 and in Tables 1 and 2.



a *b*

Figure 5. Matrixes *rij1n* $\|r_{ij}\|_1^n$ and *ωij1n* $\|\omega_{ij}\|_1^n$:
a — *rij1n* $\|r_{ij}\|_1^n$ — Matrix of connections of the neural network;
b — *ωij1n* $\|\omega_{ij}\|_1^n$ — Matrix of the relative weights of the neural network elements
S o u r c e : made by the author

Table 1

Boolean vertexes

Boolean vertex type	AND-AND	AND-OR	AND-AND/OR	OR-AND	AND/OR-AND
Sub-item number	I	II	III	IV	V

Table 2

Conventional symbols

$\lambda = 0 \rightarrow$	$\mu = 0 \rightarrow$	$\lambda \leq 1 \rightarrow$
$\mu < 1 \rightarrow$	$\lambda = 1 \rightarrow$	$\mu = 1 \rightarrow$

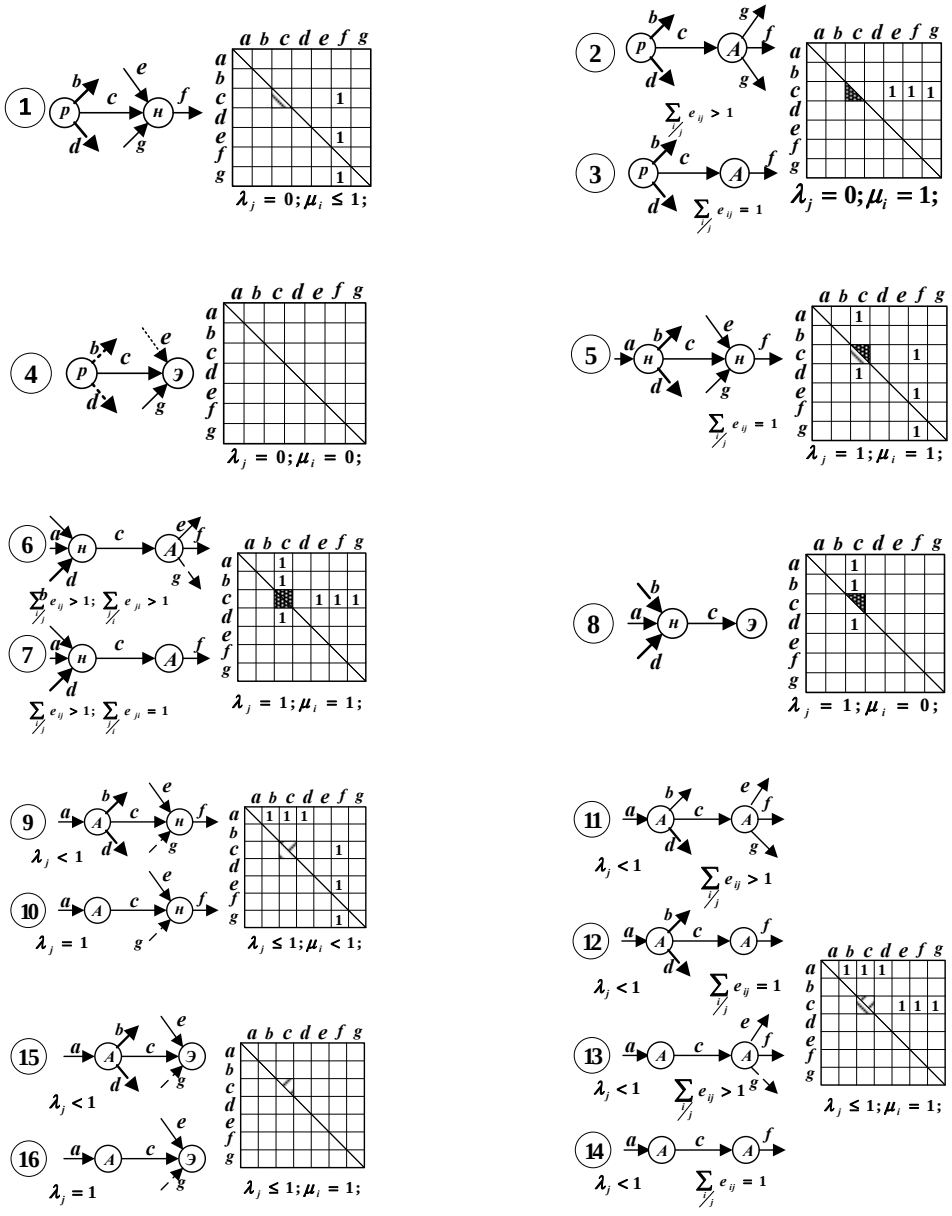


Figure 6. Sub-options of arcs connecting neural network vertices

S o u r c e : made by the author

The same options with the corresponding values $\lambda_{j=i}$; $\mu_{i=j}$; $\sum_{i=1/j} r_{ij}$; $\sum_{j=1/i} r_{ij}$ are presented in Table 3. Table 3 shows the possible types of Boolean vertices, the designation of which is in Tables 1 and 2 and in Figure 7. Taking into account the Boolean types of vertices, the total number of sub-options for neural network arcs will be 110 (see Table 3). The number of such sub-options can be much more if, in addition to Boolean vertices, certain types of vertices with restrictions, vertices with negation, or vertices whose logical functions are determined by Venn diagrams proposed for a formal neuron [24] will be introduced. The properties of a formal neuron are described in the relevant literature [24; 25], etc.

So, the main elements of neural network models can be represented using the elements (vertices and arcs) of the canonical edge graph.

Next it is necessary to consider an example of constructing a structural diagram of an automaton based on formal neurons (AFN) according to the given functions of this automaton.

Table 3

Characteristics of arcs connections

Option number	Properties of the connections				Initial vertex		Final vertex		Number of options
	$\lambda_{j=i}$	$\mu_{i=j}$	$\sum_{i/j} e_{ij}$	$\sum_{j/i} e_{ij}$	View	Boolean type	View	Boolean type	
1	0	$0 < \mu_i < 1$	(0)	(1)	P	I,II,III	H	I,IV,V	9
2	0	1	(0)	>1			A	I,II,III	9
3	0	1	(0)	1			A	I	3
4	0	0	(0)	(0)			Э	I,IV,V	9
5	1	$0 < \mu_i < 1$	>1	(1)	H	I,IV,V	H	I,IV,V	9
6	1	1	>1	>1			A	I,II,III	9
7	1	1	>1	1			A	I	3
8	1	0	>1	(0)			Э	I,IV,V	9
9	$0 < \lambda_i < 1$	$0 < \mu_i < 1$	(1)	(1)	A	I,II,III	H	I,IV,V	9
10	1	$0 < \mu_i < 1$	1	(1)		I	H	I,IV,V	3
11	$0 < \lambda_i < 1$	1	(1)	>1		I,II,III	A	I,II,III	9
12	$0 < \lambda_i < 1$	1	(1)	1		I,II,III	A	I	3
13	1	1	1	>1		I	A	I,II,III	3
14	1	1	1	1		I	A	I	1
15	$0 < \lambda_i < 1$	0	(1)	(0)		I,II,III	Э	I,IV,V	2
16	1	0	1	(0)		I	Э	I,IV,V	3

5. An example of constructing a block diagram of a finite automaton on formal neurons

As an example, let's consider the simplest cases when a neuron implements only such logical functions as: "AND" or "OR" (Figure 7, *a*), which are Boolean. They determine the θ value, the threshold of the neuron. Let the finite machine be given by a diagram (Figure 7, *b*) or Table 4. In order to describe the states of the automaton, one can apply the representation of the states of the automaton using Boolean functions, depending on the reasons that generate these states. You can then replace each of the Boolean expressions containing the same operations with a single character.

Let us compile a Table 5, in which we assign the causes to each new state or output signal which was generated by them in the form of strict disjunctions of the intersections of input signals and current states. For example, in order for the automaton to transfer to the A state, it is necessary that the input of the machine in state B be given a signal a, or that the input of the machine in state C be given signal b. Next let us compose the initial set, which includes all states, all input signals, as well as all necessary combinations of current states and input signals, a set of states, a set of input signals, input and output blocks of the automaton.

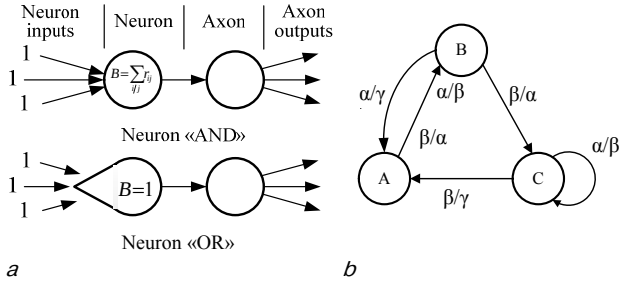


Figure 7.
a — The simplest cases when a neuron implements only such logical functions as: "AND" or "OR";
b — Graph implementation of the finite machine

Table 4

Diagram of the finite machine

Initial (current) state	A		B		C	
Initial symbol	a	b	a	b	a	b
Next state	B	B	A	C	C	A
Output symbol	β	α	γ	β	α	γ

Table 5

States and symbols of the finite machine and their previous combinations

State and symbol of the automata	Previous combination of states and symbols
a	—
b	—
A	$B \cap a \oplus C \cap b$
B	$A \cap a \oplus A \cap b$
C	$B \cap b \oplus C \cap a$
α	$A \cap b \oplus B \cap b$
β	$A \cap a \oplus C \cap a$
γ	$B \cap a \oplus C \cap b$

Table 6

Designation of both alphabetic and numeric characters and their combinations

Elements of the initial set		Previous combinations of elements of initial states
Input device	X	
The set of the initial states	X_1	X
The set of the input states	X_2	X
Input symbols	a	X_2
	b	X_2
The states of the automata	A	$X_1 \oplus d \oplus e$
	B	$X_1 \oplus f \oplus g$
	C	$X_1 \oplus h \oplus k$
Combinations of current (initial) or next states and input symbols	d	$B \cap a$
	e	$C \cap b$
	f	$A \cap a$
	g	$A \cap b$
	h	$B \cap b$
	k	$C \cap a$
Input symbols	α	$g \oplus h$
	β	$f \oplus k$
	γ	$a \oplus e$
The output of the devices	Y	$\alpha \oplus \beta \oplus \gamma$

Each of the elements of this set will be assigned an alphabetic or numeric symbol (Table 6). For each element of the second column of Table 6 in the third column we shall indicate those elements that precede the elements of the second column or generate them.

The diagram of the machine under consideration includes six combinations of current states and input signals. Let us designate these combinations with letters: d, e, f, g, h, k. Now we can consider the current state of B and the input signal as causes that give rise to d element, and so on. In turn, the state B is generated either by choosing this state from the set X_1 of all initial states, or by the f element, or by the g element. Table 6 makes it possible create a matrix of binary relations defined on the original set or an adjacency matrix of elements of the original set (Figure 8). The same figure shows tables of logical functions of neurons and axons.

Obviously, each neuron will be formed from a column that has more than one $l_{ij} = 1$ element. In turn, each row of the $\|l_{ij}\|_1^n$ matrix, which has more than one $l_{ij} = 1$ element, will allow one axon to be formed. Axons will also be formed from those rows in which there will be one $l_{ij} = 1$ element, if this element is not included in the column from which the neuron will be formed.

The tables of neurons and axons also indicate the logical functions they implement. The functions of neurons are determined directly from Table 6. The outputs of axons X, X_1 and X_2 are determined by the fact that at the same time the machine can be in only one of the three states, and only one of the two signals can be applied to its input. In other cases, axons (in the machine under consideration) transmit the same signal generated by the corresponding neuron through all outputs. For example, the axon defined by the $i = a$ string simultaneously transmits a signal a to d, f and k elements.

The matrix $\|l_{ij}\|_1^n$ is normalized. In this case, the cyclomatic number of the graph remains unchanged. In the $\|l_{ij}\|_1^n$ matrix, the $l_{ij} = 1$ elements that are subjected to the Δn — transformation are circled. Let us denote these elements as $\hat{l}_{ij}$. The condition of conservation of the cyclomatic number allows us to immediately calculate all the main characteristics of canonical graphs (Figure 8):

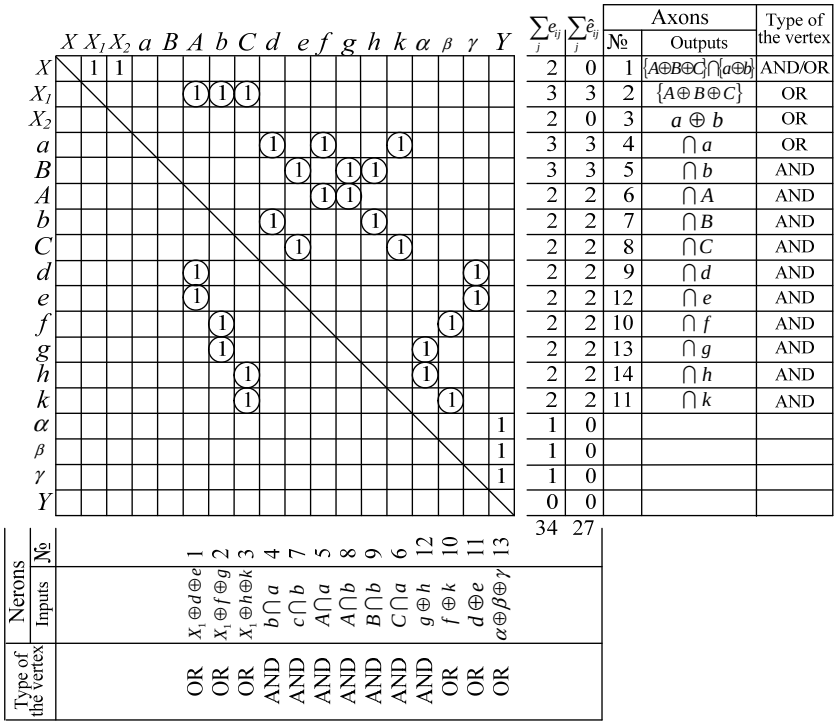


Figure 8. An adjacency matrix of the elements of the original ($n_0 = 18$) set. Logical functions of the neurons and the axons
S o u r c e : made by the author

- The order of the matrix:

  $n_k = n_0 + \Delta n = 18 + 27 = 45.$

- The cyclomatic number:

$$\nu(H_k) = \nu(G) + \sum_i \sum_j e_{ij} - n_0 + 1 = \\ = 34 - 18 + 1 = 17.$$

- The number of arcs of the canonical edge graph: $\nu(\bar{Q}_k) = n_k = 45.$
- The number of the vertexes of the canonical edge graph:

  $\nu(V_k) = n_k - \nu(H_k) + 1 = 45 - 17 + 1 = 29.$

- The number of the arcs of the canonical edge graph:

$$\nu(\Gamma_k) = \sum_i \sum_j e_{ij} + \sum_i \sum_j \hat{e}_{ij} = 34 + 27 = \\ = 61.$$

- The number of the vertexes of the canonical vertex graph: $\nu(Q_k) = n_k = 45.$

The normal matrix $\left\|\bar{r}_{ij}\right\|_1^n$ of the finite machine neural network is shown in Figure 9. Table 7 summarizes the characteristics of the arcs of the canonical edge graph with their initial and final vertices as elements of a neural network of a finite machine. The construction of a block diagram of a finite automaton by the $\left\|\bar{r}_{ij}\right\|_1^n$ matrix (Figure 9) and Table 7 is easy. Scheme in the form of an edge graph is shown in Figure 10. It is easy to verify that this circuit exactly performs the functions of a given finite state machine.

Let us set the automaton represented by the diagram in Figure 7, b, initial state A and input signal program $ababab$ ...

Then the alternation of new states and output signals will be as follows:

- Initial and current state of $ABCCAB$ ...;
- $ababab$... inputs;
- Following states $BCCAB$ ...;
- Output signals $\beta\alpha\beta\gamma\beta$ ...

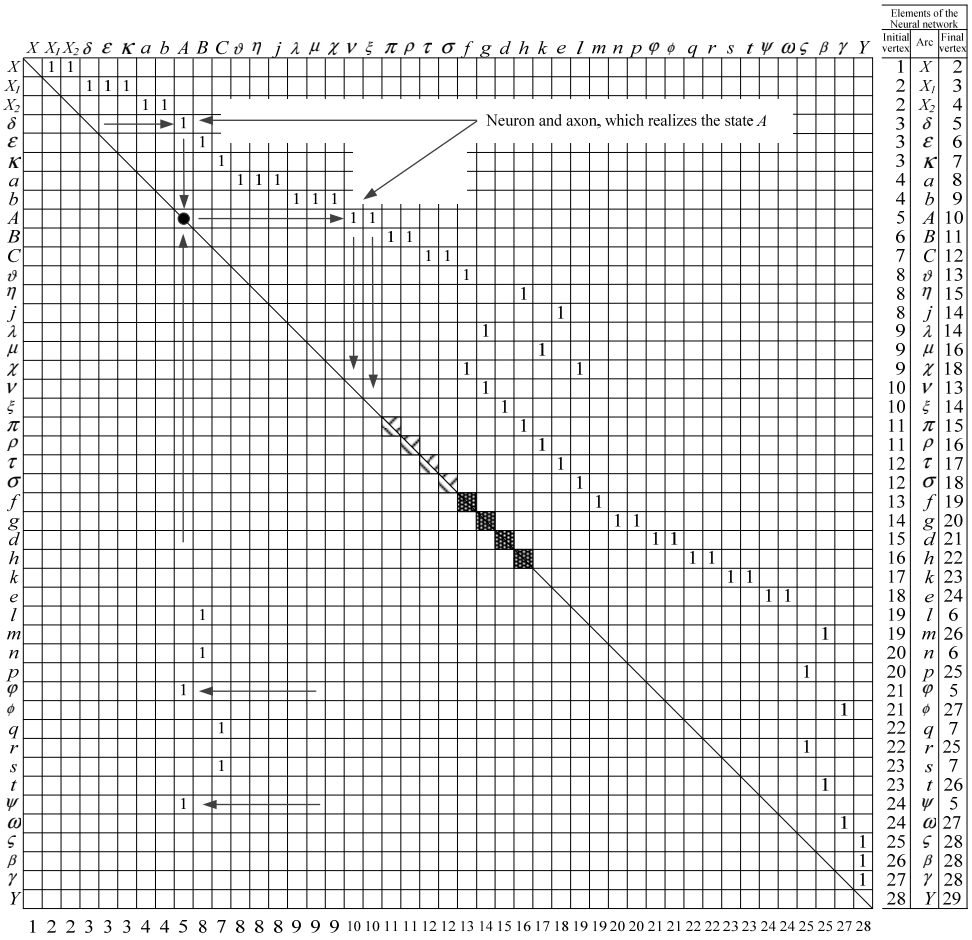


Figure 9. The normal matrix $\|\bar{r}_{ij}\|_1^n$ of the finite machine
S o u r c e : made by the author

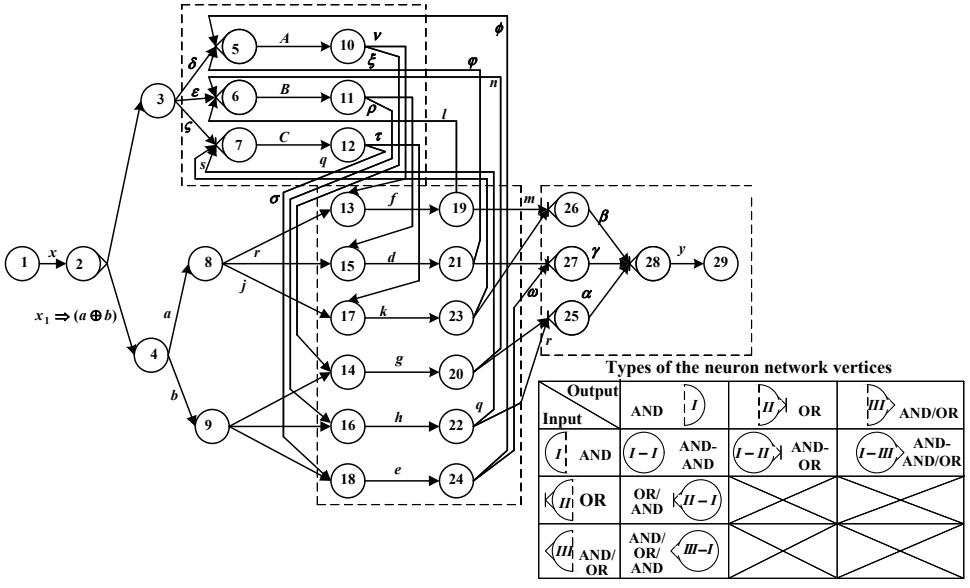


Figure 10. The graph of a finite automaton
S o u r c e : made by the author

Table 7

The characteristics of the arcs of the canonical edge graph with their initial and final vertices as elements of a neural network of a finite machine

Elements of the finite machine	Initial vertex		Purpose of the finite state machine element	Final vertex	
	Type	Input function		Type	Output function
Input device. Sensors of initial states and input symbols	I	START	The task of start and the work of the program	III	$X_1 \cap X_2$
	III	$X_1 \cap X_2$	Setting initial states	II	$A \oplus B \oplus C$
			Setting input symbol	III	$a \oplus b$
Signal transmission operations	II	$X_1 \Longrightarrow A \oplus B \oplus C$	Passing command to state A	IV	A
			Passing command to state B	IV	B
			Passing command to state C	IV	C
Neurons — Axons	II	$X_2 \Longrightarrow a \oplus b$	Passing input symbol a	I	$\cup a$
			Passing input symbol b	I	$\cup b$
	IV	$A \oplus d \oplus e$	State implementation A	I	$\cup A$
		$B \oplus f \oplus g$	State implementation B	I	$\cup B$
		$C \oplus h \oplus k$	State implementation C	I	$\cup C$
Signal transmission operations	I	a	Passing input symbol a	I	f
					d
					K
	I	b	Passing input symbol b	I	g
					h
					e
	I	A	Passing command: current state A	I	f
					d
	I	B	Passing command: current state B	I	K
					g
	I	C	Passing command: current state C	I	h
					e
Neurons — Axons	I	$A \cup a$	The formation of f signal	I	$\cup f$
		$A \cup b$	The formation of g signal		$\cup g$
		$B \cup a$	The formation of d signal		$\cup d$
		$B \cup b$	The formation of h signal		$\cup h$
		$C \cup a$	The formation of K signal		$\cup K$
		$C \cup b$	The formation of e signal		$\cup fe$
Signal transmission operations	I	f	Passing signal f	IV	β
					β
	I	g	Passing signal g		B
					d
	I	d	Passing signal d		A
					γ
	I	h	Passing signal h		C
					α
	I	K	Passing signal K		C
					β
	I	e	Passing signal e		A
					γ
Neurons — Axons	IV	$g \oplus h$	The formation of the output symbol α	IV	$\alpha \oplus \beta \oplus \gamma$
		$f \oplus K$	The formation of the output symbol β		
		$d \oplus e$	The formation of the output symbol γ		
Output device	IV	$\alpha \oplus \beta \oplus \gamma$	Passing of the output symbol	I	Y

Let us check the operation of the neural network according to the given program.

1. The input device X sets the program of work for the sensor of initial states X_1 and the sensor of input X_2 signals.

2. The sensor of initial states X_1 generates a command to transfer the automaton to A state.

3. The input signal of sensor X_2 generates input signals according to the program specified by the input X device.

4. The command to switch the automaton to state A is transmitted along the arc 3→5 to neuron 5, which implements the specified state on the arc A $(5 \to 10)$ at the t_0 moment. The signal about this, equal to 1, is transmitted at the moment $t_0 + 1$ to vertices 13 and 14.

5. From vertex 4, the signal $|a| = 1$ is transmitted along arcs 4→8 and 8→13, 8→15 and 8→17 to vertices 13, 15 and 17.

6. Vertex 13 is the initial vertex of the neuron "AND", the threshold of which is equal to: $\theta = 2$. Since two signals come to this vertex, neuron 13 generates a signal: $f = A \cap a$, which is transmitted at the moment $t_0 + 2$ to vertices 6 and 26.

7. When the signal $|f| = 1$ enters vertex 6, this neuron implements state B, and at time $t_0 + 3$ the axon sends a single signal about this to vertices 15 and 16. At the same time, neuron 26 generates a β signal, which at time: $t_0 + 3$ is transmitted to the output Y device.

8. Since the input signal a is implemented at time: $t_0 + 2$, then the element X_2 at time: $t_0 + 2$ generates signal b, which is transmitted to vertexes 14, 16 and 18.

9. At vertex 16, the sum of the input signals is equal to the threshold, so neuron 16 generates an h signal, which at time: $t_0 + 3$ is transmitted from vertex 22 to vertices 7 and 25.

10. Neuron 7 implements Cstate and neuron 25 generates α signal.

Further operation of the machine is evident and does not require explanation. It is clear that the obtained scheme can be completely replaced by a real construction, including certain physical models of formal neurons. The above example shows the synthesis of a very simple machine, but the same general principles can be applied to synthesise other, more complicated machine.

Conclusion

1. The structure of the basic elements of AFN allows their convenient graphical representation in the form of basic elements of canonical edge graphs. At the same time, the representation of the AFN elements, in which logical functions are implemented, is provided by means of those elements of the canonical edge graph, in which logical functions are also implemented.

2. The similarity of the structures of canonical edge graphs and automation on formal neurons (AFN) allows building block diagrams of AFN automatically, provided that the sets of the states, the input and output signals of the original finite machine are specified in the form of a finite vertex graph.

3. The implementation of Δn — transformation and matrix normalization make it possible to arrange the DFSM using a linear algorithm, and sometimes polynomial in complexity.

4. The main advantage of this approach is that in its implementation the representation of complicated logical functions does not require the use of polynomial algorithms for programming.

5. Abandoning polynomial algorithms for the representation of logical functions will eventually lead to a decrease in energy costs and an increase in accuracy in their calculation. In terms of chip and printed circuit board technologies, this can lead to a reduction in chip size and thickness.

References

1. Minsky M. *Computation. Finite and infinite machines*. Prentice Hall International, 1972.

2. McCallouch WS, Pitts W. A Logical Calculous of the Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics.* 1943;5:115–133. https://doi.org/10.1007/BF02478259

3. Barkalov A, Titarenko L. *Logical synthesis for FSM-based Control Units*. Lecture Notes in Electrical Engineering. Springer Sience& Business Media; 2009. http://doi.org/10.1007/978-3-642-04309-3

4. Bordihn H, Holzer M, Kutrib M. Determination of finite automata accepting subregular languages. Giessen: Elsevier, 2009.

5. Lamberti G, Scandale M. Incremental determinezation and minimization of finite acyclic automata. *IEEE International Conference on Systems, Man, and Cybernetics.* Manchester, UK, 2013;2250–2257. https://doi.org/10.1109/SMC.2013.385

6. Gandhi A, Ke NR, Khoussainov B. Descriptional complexity of determinization and complementation for finite automata. *Proceedings of the Seventeenth Computing: The Australasian Theory Symposium.* Australia, 2011; 119:95–104.

7. Buchsbaum AL, Giancarlo R, Westbrook JR. On the determination of weighted finite automata. *Sosiety for*

Industrial and Applied Mathematics. 2000;30(5):1502–1531.

8. Shalyto A. Logic Control and "Reactive" systems: Algoritmization and programming. *Automation and Remote Control.* 2001;1:1–29. (In Russ.)

9. Vinogradova M, Tkachev S, Kandaurova I. The determining finite automata process. *Mathematics and Mathematical Modeling.* 2017;4:1–17. (In Russ.) https://doi.org/10.24108/mathm.0417.0000067

10. Gorachkin B. The development of the theory of finite automataand its applications. *Engineering bulletin.* 2015;4:538–542. (In Russ.) EDN: TVWZNT

11. Verevkin A, Kiryushin O. The Synthesis of Complex Logical Controllers with Variables of Boolean and Fuzzy Logics. *Proceedings of the 7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019).* Ufa: Atlantis Press; 2019. https://doi.org/10.2991/itids-19.2019.9

12. Burdonov IB, Kosachev AS, Kulyamin VV. The use of finite automata for testing programs. *Programming.* 2000;2:12–28. (In Russ.)

13. Kleene S. Introduction to metamathematics. *Bull. Math. Biophys.* 1943;5:115–133.

14. Godel K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik.* 1931;I(38): 173–198. https://doi.org/10.1007/BF01700692

15. Turing A. Can a Machine Think? The World of Mathematics Universal Turing Machine. *The world of Mathematics.* 1956;4:2109.

16. Post E. Formal Reductions of General Combinatorics Desision Problem. *American Journal of Mathematic.* 1943;65(2):197–215.

17. Mitchell M. *An introduction to the genetic algorithms*. London: MIT Press Cambridge, Massachusetts; 1999.

18. Fogel L, Owens A, Walsh MJ. *Artifisial Intellegence through Simulated Evolution*. NY: Wiley; 1966.

19. Bukatova I. *Evolutionary Modelling and its Applications.* Moscow: Nauka Publ.; 1979. (In Russ.)

20. Zaikin AK. Development of finite automata creation methods with annealing simulation algorithm by the "War for resources" example. *Scientific and technical journal of information technologies, mechanics and optics.* 2011;2(72):49–54. (In Russ.) EDN: NECKCX

21. Harman M, Mansouri A, Zhang Y. Search-Based Software Engineering: A Comprehensive Analysis and Review of Trends, Techniques, and Applications," Dept. of Computer Science. London: King's, 2007.

22. Rosenblatt F. *Principles of neurodynamics.* Buffalo: Cornell Neurotical Laboratory, 1965.

23. Malinin LI, Malinina NL. *On the solution of Graph Isomorphism*. 2022. (In Russ.) Available from: https://www.researchgate.net/publication/358570634_On_the_solution_of_ the_Graph_Isomorphism_Problem.

24. Arbib MA. *Brains, machines and mathematics*. McGraw-Hill, 1964.

25. Nechiporenko V. *Structural analysis and methods for building reliable systems.* Moscow: Sovetskoe Radio, 1968. (In Russ.)

About the author

Natalia L. Malinina, Candidate of Physical and Mathematical Sciences, Associate Professor of the Department 604, Aerospace Faculty, Moscow Aviation Institute (National Research University), Moscow, Russian Federation; ORCID: 0000-0003-0116-5889; malinina806@gmail.com