DOI: 10.51790/2712-9942-2020-1-2-6

B-COMPUTERS: SELF-EVOLUTION

Georgiy E. Deev[a], Sergey V. Ermakov[b]
Obninsk Institute for Nuclear Power Engineering, National Research Nuclear University MEPhI, Obninsk, Russian Federation
[a] georgdeo@mail.ru, [b] ermakov@iate.obninsk.ru

Abstract: the study demonstrates that the evolution of the B-computer can be unlimited as it can integrate a range of other digital devices. With this fact, we claimed that AI can be implemented with B-computers.

Keywords: B-computer, artificial intelligence (AI), PCB layout template, abstract computer (AC), B-circuit.

Acknowledgements: I wish to show my appreciation to A.V. Lamkov for reviewing this paper. This study is supported by RFBR grant 20-07-00862.

Cite this article: Deev G. E., Ermakov S. V. B-Computers: Self-Evolution. *Russian Journal of Cybernetics*. 2020;1(2):47–55. DOI: 10.51790/2712-9942-2020-1-2-6.

B-КОМПЬЮТЕРЫ: САМОРАЗВИТИЕ

Г. Е. Деев[а], С. В. Ермаков[б]
Обнинский институт атомной энергетики, Национальный исследовательский ядерный университет «МИФИ», г. Обнинск, Российская Федерация,
[а] georgdeo@mail.ru, [б] ermakov@iate.obninsk.ru

Аннотация: показано, что B-компьютеры обладают неограниченным потенциалом развития, реализуемым на уровне разнообразных цифровых устройств, интегрируемых B-компьютером. На основании этого факта делается вывод о том, что решение задачи создания искусственного интеллекта в полной мере может быть осуществлено на базе B-компьютеров.

Ключевые слова: B-компьютер, искусственный интеллект (ИИ), шаблон заготовки, абстрактное вычислительное устройство (АВУ), B-схема.

Благодарности: авторы выражают благодарность А. В. Ламкову за обсуждение работы. Работа выполнена при поддержке гранта РФФИ 20-07-00862.

Для цитирования: Деев Г. Е., Ермаков С. В. B-компьютеры: саморазвитие. *Успехи кибернетики*. 2020;1(2):47–55. DOI: 10.51790/2712-9942-2020-1-2-6.

A Preliminary Informal Explanation

This paper refers to “A-computers” and “B-computers” (the letters “A” and “B” are Latin). By A-computers we mean computers and related devices such as all kinds of gadgets. In conjunction with A-computers are B-computers. The A-computers have a historical priority, they are primary, which is emphasized by the use of the letter “A” in their name. Concerning them, B-computers are secondary, which is also shaded by the use of the letter “B” in their name. A-computers and B-computers unite and at the same time contrast related principles of abstract schemes construction. Principle A, which serves to build A-computer circuits, states: *different information signals can be transmitted over the same line.* But, understandably, too many different information signals cannot be transmitted over the same row, because in this case there is a difficult problem of distinguishing signals, leading to unreliable calculations. This was the reason for J. von Neumann to formulate the thesis that the best system of notation suitable for computers is binary notation. And indeed in this case the calculations are as reliable as possible. The whole computer industry has been binary ever since. A-computer computing, while being as reliable as possible in the binary number system, is nevertheless not free from failures.

Principle A is contrasted with its dual principle B, which states that different information signals cannot be transmitted along the same row. As we can see, both principles, being the antithesis of each other, complement one another so that there is no third. Thus, under Principle B, there is a one-to-one mapping

between the row and the information signal, and there is never any confusion of signals in the row. This ensures the reliability of calculations for any number of signals. Therefore Neumann's thesis does not apply to B-computers. Consequently, B-computers can be created for any number system without experiencing any difficulties in principle. One of the features of B-computers is the simplicity of the basis on which both B-computers themselves and the B-devices contained in them can be built. This structural basis consists of two elements: the row, which transmits information signals, and the element &, which controls the direction of the information signal. The basis can be represented by the notation: { | , & }; this set of structural elements satisfies the principle of maximum simplicity. Consequently, B-circuits of B-computers are as simple as possible in their design.

Every object is at some stage of development. The stage of development of an object is understood as its constructive state, as well as a set of functional interactions of the object with the environment, which we will call a set of functions of the object. By development, we mean the transition of an object from one stage of development to another. This implies both a possible constructive change of the object itself and a change in the set of functions associated with it. In connection with the concept of development, it is immediately necessary to characterize this development, whether the development will be regressive or progressive. As stated above, each stage of development is characterized by two components: the structural features of the object and its function set. It is quite natural to assume that the functional set of an object depends on its construction so that the set of functions characterizing a stage of development is a kind of function of its constructive perfection, which is reflected by the notation $\{f_1,f_2,\dots\} = \Psi(k)$, where k is the coefficient characterizing constructive development. Let us assume that it varies within the limits $0 \le k < \infty$ (there is no limit to perfection). Thus the stage of development of an object is a pair $\langle k,\{f_1,f_2,\dots\}\rangle$, consisting of the coefficient of constructive development k and a set of functions $\{f_1,f_2,\dots\}$. This is a peculiar vector of development. Let us not, however, further deepen the vector analogy. Both regression and progress are characterized by the behavior of the pair $\langle k,\{f_1,f_2,\dots\}\rangle$. Both factors included in this pair may be determined by external influence, in which case we speak of induced development. But it may also be the case that the behavior of the pair $\langle k,\{f_1,f_2,\dots\}\rangle$ is determined by internal causes inherent in the object, with little or no external influence. Then we speak about *self-development.*

Closely related to the concept of self-development is the concept of *artificial intelligence* (AI), so popular recently. *Artificial intelligence is understood as an artificially created object, which has the ability to independent intellectual activity.* Thus, the first distinguishing feature of artificial intelligence is its artificial origin, i.e. it must be created by someone else capable of creative activity, such as a human creator or some other thinking being (MC) capable of creating creations. For this reason, a fly is not artificial intelligence, even though it shows signs of intelligence. It is not for the reason that neither man nor any other MS is its creator. On the other hand, a car that rolls around on the floor and is controlled by a boy with levers on a remote control is also not an AI, despite the apparent intelligence of its behavior. Its apparent intelligence is the result of the boy's intelligence, not the machine itself. Similarly, the famous Japanese robots are not examples of AI, because although they are artificially created, they do not have an intelligence of their own; they behave by obeying orders from computers embedded in them, which in turn contain programs written by humans. These programs determine the robots' behavior. Ultimately, the robot's behavior is determined by humans, just as in the case of a machine. All robots are controlled by A-computers, an attribute of which are programs created by humans. Therefore, the behavior of robots is entirely dependent on humans. Consequently, as long as they are controlled by A-computers, no "machine revolt" is possible. Simply because these machines have no intelligence of their own.

B-computers are another matter. B-computers have unlimited development potential. The presence of developmental potential allows the possibility of self-development and, eventually, its transformation into intelligence. This potential can be used by humans for their purposes, or it can be used by the computer itself for its development. By acquiring intelligence, the B-computer can enter into a competitive intellectual game with humans and even win it. Not necessarily, however, it must end in "war", a friendly symbiosis of man and machine is quite possible.

Synthesis of Fully Defined Devices on Blank Templates

Blanks are abstract boards on which semi-finished products for future device circuits are prepared in advance. The blanks meet the requirement of maximum simplicity. On the blanks, there are rows for

transmitting information signals, as well as many elements &. There is nothing else. If all rows in the template are parallel, then in such a template it is not possible to vary the links. Variation of links is only possible in a billet where there are intersecting rows. If there are intersecting rows, then their connection, although possible, is not necessary. The presence of a connection will be represented by a bold dot in the crosshair, as shown in Fig. 1.

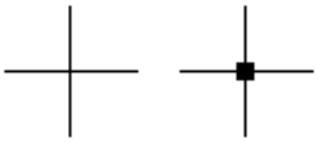

Figure 1. *No connection (left), connection (right)*

Thus, in abstract B-schemes, the connection operation is reduced to the simplest operation – a point.

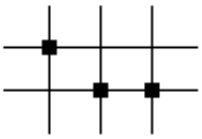

Figure 2. *An example of the connection option*

Figure 2 shows one possible way of connecting the five crossed rows. There are a total of 2^6 possible variants. This number evaluates the development potential (potential of possibilities) of this configuration of rows.

On the next page is an example of a typical template. On the left is a "raw" template with no connections; on the right is the same template, but with one of the possible connections selected. There are unimaginably many connection options (2^{728}), and each connection option corresponds to a different computing device. Not all of them are of any interest at the moment, but some certainly are. Such is the device resulting from the shown connection variants. This device is nothing other than a multiplier by three in the quadratic number system. They are denoted as: $\bar{x} \cdot 3|\bar{q}$, where $\bar{x}$ is the numberid written on the grid $Gr^0 = \ldots \overline{2}\,\overline{1}\,\overline{0}$representing, in particular, natural numbers; $\bar{q}$ is the state of the multiplier $\bar{q} \in \{0,1,2\}$. As you can see, after selecting a connection option, "extra" rows may remain on the template. Not playing any role concerning the chosen connection variant, they can carry new information signals and be useful for other purposes. The number 2^{728} expresses the "hidden" potential for development, the potential for opportunities associated with this type of template. If we consider that there can also be unimaginably many types of blanks, we can imagine what is the potential for development associated with B-computers. The symbolic attributes present in Fig. 3, is necessary for us to understand the functioning of the device. The device itself, at its current stage of development, does not need it. Among the many modes of operation of the device, there are modes meaningful from our point of view. Therefore, it makes sense to equip some elements of the circuit with informational signals that we understand. Thus, symbols 0, 1, 2, 3 are provided with some rows of the device. Groups of four rows form input and output devices, groups of three rows - control signals. The group of elements & , with 0 on the first (left) place, forms the state $\bar{q} = 0$; similarly, the states $\bar{q} = 1$ and $\bar{q} = 2$.

The state $\bar{q} = 0$ will be called the initial state. The device performs multiplication by three by being put into the initial state $t = 0$. If it is put in other states for $t = 0$, it calculates other functions.

Let's look at an example of calculation with this device.

Example 1. Let us assume that $\bar{x} = \ldots 00322$ is in the quadratic number system. We follow the computation using the time sweep of the computation process.

The information in the sweep is read from right to left. In the same way, the numbers of the digits in the numeric digits grow from right to left. The first row shows the moments in which the device functions. These moments are non-negative integers. In the second row are the digits of the number applied to the input. In the bit 0 corresponding to the moment of $t = 0$, there is a digit $x_0 = 2$. Similarly, in all other

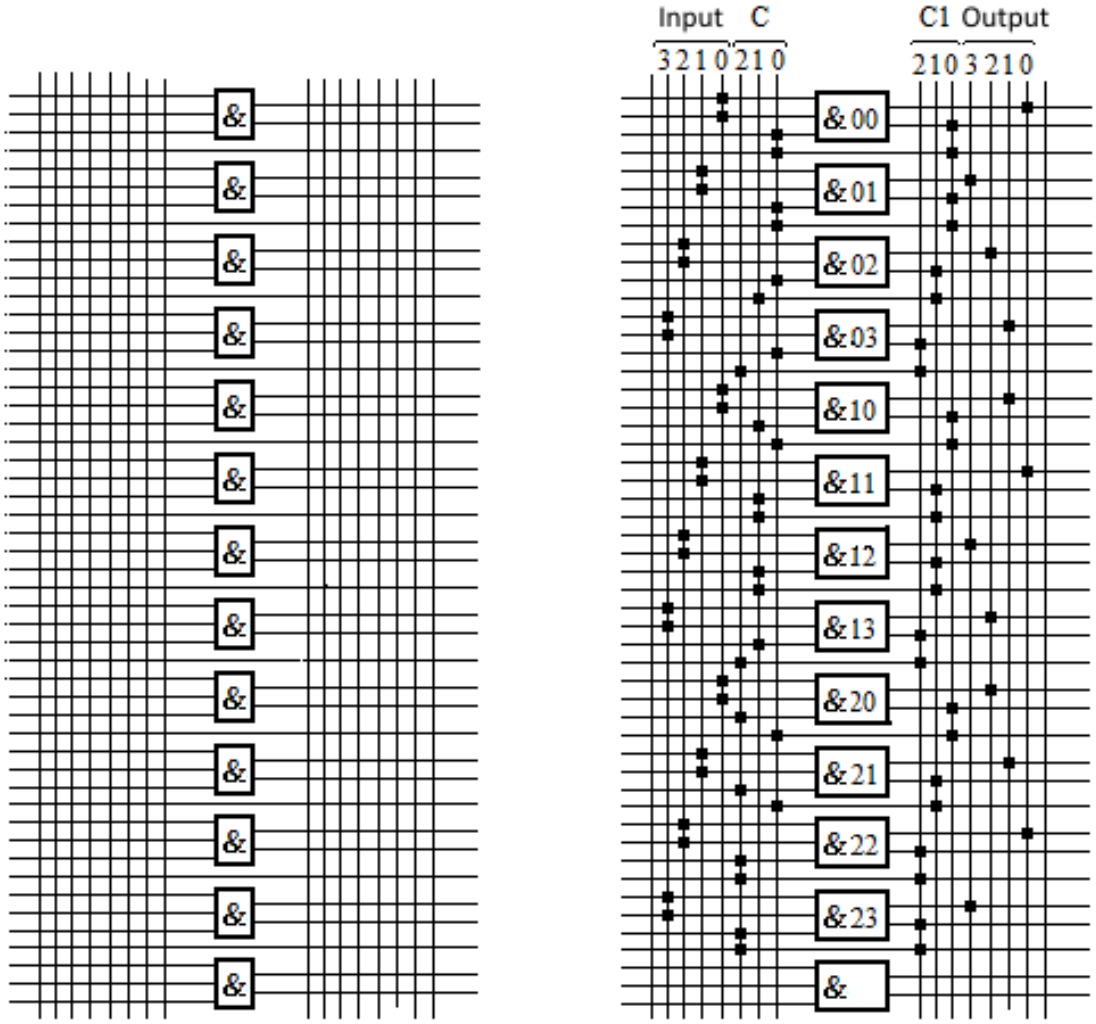

Figure 3. *Template without connections (left) and template with connections (right)*

6	5	4	3	2	1	0	t
...	0	0	0	3	2	2	x_t
&00	&00	&00	&20	&13	&12	&02	$\&qx_t$
...	0	0	2	2	3	2	y_t

bits. In the row, $\&qx_t$ there are elements of &, which are triggered at time t. The row y_t contains bitwise added digits of the result, and the correspondence between the moment t and the digit number is preserved, just as for the input. Note that the device is an abstraction, i.e., an ideal object in which everything is perfectly actuated, all elements & have the same delay; the output signal is output at the same time moment as the incoming input. In reality, the output signal is always output with some delay relative to the input signal. But computational engineers always try to reduce this delay as much as possible. Mathematicians take this process down, reducing the lag to zero. Clearly, this is an idealization, but it does not spoil the mathematical content of the computational process, rather it represents it.

The design of the device alone is not enough to make the principle of its operation clear. It is required to give the law of functioning of the device, which is provided by its functioning equations. They have the form:

$$\begin{cases} \bar{q}_{t+1} = \theta\bar{q}_t x_t = \rangle 3 \cdot x_t + \bar{q}_t \langle , \\ y_t = \sigma\bar{q}_t x_t = \langle 3 \cdot x + \bar{q} \rangle , \end{cases} \tag{1}$$

where $\bar{q} \in Q_3 = \{0, 1, 2\}$, $x, y \in Z_{(4)} = \{0, 1, 2, 3\}$ θ is the function of transitions, σ

is the function of outputs. In (1) we use the notations: $\langle\bar{\alpha}\rangle = \langle\ldots 00\alpha_{\text{r}}\ldots\alpha_1\alpha_0\rangle = \alpha_0$ and $\rangle\bar{\alpha}\langle = \rangle\ldots 00\alpha_{\text{r}}\ldots\alpha_1\alpha_0\langle = \ldots 00\alpha_{\text{r}}\ldots\alpha_1$.

Read the sweep: for $t = 0$ the device is in the initial state $\bar{q} = 0$, which is provided by the arrival of the control signal on row-0 of input-C. All four elements & of the initial state are ready to be triggered: &00, &01, &02,&03. But only one of them will actuate, namely, the one which corresponds to the digit received at the input, which is the digit $x_0 = 2$. That is why the element &02, which is marked in the sweep, is triggered. According to the device diagram, digit 2 appears on the output, and the control signal for the moment $t = 1$ is transmitted to elements & state 1, i.e. to the elements &10, &11, &12, &13. At the same moment, the digit $x_1 = 2$ is at the input; then it goes similarly to $t = 0$. As a result, the output will form the result, the number $\bar{y} = \ldots 002232$.

Let us write down the result of the device in the form: $\ldots 00322\cdot 3|0 = \ldots 002232$.

Read: if we put the device in the initial state $\bar{q} = 0$ (marked by zero to the right of the vertical row) and input the number $\bar{x} = \ldots 00322$, then the device will output the result $\bar{y} = \ldots 2232$, which is the product of the number $\bar{x} = \ldots 00322$ by three: $\ldots 00322\cdot 3 = \ldots 2232$. The result is checked by the school multiplication "column" (in the quadratic notation system).

Note that the result is output at $t = 2$, i.e., in 3 (not 4) clock cycles. This is explained by the fact that the highest two digits of the result are output by the device at the same time. This phenomenon is generalized when considering all B-devices, which leads to the conclusion that in B-devices the computation ends at the moment when the highest digit comes from the input. This is only true for B-devices. Similarly, we can parse the operation of any of the devices 2^{728} that can be created on the template pattern shown in Fig. 3. If we take into account that there are unimaginably many templates like the above and other much more complex ones, then, again, it is clear how great the potential for B-devices development is.

In the example above it was assumed that the mapping realized by the device is fully specified, i.e. either the equations of functioning of the device (1), or its table specification (was not given), or its B-scheme (was not given) are known. It was required to obtain its template B-scheme by making the necessary connections.

This is not the case in the next section, where the mapping is set in fragments.

Synthesis of devices from fragments of mappings

We will not describe the *algorithm for the synthesis of abstract computational devices (ACD) by fragments of mappings*, but only give an example of the result of its work. Hereinafter, instead of "synthesis algorithm" we will simply say "algorithm".

Let us denote ${}_pA_q$ by the symbol an abstract computational device (ACD) with anticipation for p steps and with a consequence of q steps, i.e. a device whose functioning is described by equations:

$$\begin{cases} \bar{q}_{t+1} = \theta\bar{q}_t x_{t+p}\ldots x_{t+1}x_t x_{t-1}\ldots x_{t-q}\,, \\ y_t \;\; = \sigma\bar{q}_t x_{t+p}\ldots x_{t+1}x_t x_{t-1}\ldots x_{t-q}\,, \end{cases} \quad (t\in Z=\{\ldots 2.1.0,-1,-2,\ldots\}) \tag{2}$$

where $x \in Z_k = \{0,1,\ldots,k-1\}$, $y \in Z_l = \{0,1,\ldots,l-1\}$ are the sets of digits in the k and l number systems, respectively; $k,l \geq 2$, natural numbers; $p,q \geq 0$. We denote by the symbol ${}_p\mathcal{A}_q$ the *set* of all ACDs with p step anticipation and a $q$step consequence: ${}_p\mathcal{A}_q = \{{}_pA_q\}$. The set of all ACDs is the set $\mathcal{A}=\bigcup\limits_p\bigcup\limits_q {}_p\mathcal{A}_q$. Abstract automata form the set ${}_0\mathcal{A}_0$.

A mapping fragment is the set of mappings of the type $\bar{x}\to\bar{y}$, where $\bar{x}$ is taken from the mapping definition domain and $\bar{y}$ is taken from the value domain. The number of mappings in the fragment can be any, except zero.

A mapping fragment is called representative if it allows the mapping to be found completely.

Let the numerical object $\bar{x} = 1300122203\underset{0}{1}$ expressed in the quaternary notation. It means all its digits belong to the set of digits $Z_{(4)} = \{0,1,2,3\}$. The digit numbers grow from right to left, starting from the zero digit marked with a zero from below. Let, similarly, an object $\bar{y} = 4403212103\underset{0}{4}$ be expressed in the pentatonic notation, $Z_{(5)} = \{0,1,2,3,4\}$ and let the mapping be such that there a mapping between the objects $\bar{x}$ and $\bar{y}$ which we will write on the digit grid:

10	9	8	7	6	5	4	3	2	1	0	t
1	3	0	0	1	2	2	2	0	3	1	x_t
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
4	4	0	3	2	1	2	1	0	3	4	y_t

(3)

The question is: whether there are ACDs from the class ${}_p\mathcal{A}_q$, $(p,q \geq 0)$ that implement the mapping of numerical objects $\bar{x} \to \bar{y}$?

The algorithm starts searching for an ACD from the class ${}_0\mathcal{A}_0$, i.e., first, it asks whether there exists an abstract automaton that implements the given mapping (recall that abstract automata fill the class ${}_0\mathcal{A}_0$) in the set of all ACDs.

Applying the algorithm, it turns out that there are two minimal automata A_1 and A_2 with two states, for which the mapping (3) takes place. These automata are defined by tables:

A_1:

q \ x	0	1
0	0, 0	0, 3
1	1, 4	1, 2
2	1, 1	0, 2
3	0, 4	0, 3

A_2:

q \ x	0	1
0	1, 3	0, 0
1	1, 4	0, 2
2	1, 1	0, 2
3	0, 4	1, 3

(4)

In the tables $x \in Z_{(4)} = \{0,1,2,3\}$, q is the state of the automaton, $\bar{q} \in Q = \{0, 1\}$, Q is the set of states of the automaton. At the intersection of row-x and column-q there is a pair (θ,σ), where $\theta = \theta(q,x)$ is the value of the transition function on the pair (q,x), and $\sigma = \sigma(q,x)$ is the value of the output function on the pair (q,x).

Thus, according to the algorithm, there are two automata mappings 1 and 2, realized, respectively, by automata A_1 and A_2, for each of which the given correspondence takes place $\bar{x} \to \bar{y}$.

Let us check that this is true. Pass the object $\bar{x}$ through automata A_1 and A_2 and make sure that in both cases the output is represented $\bar{y}$. Let us plot the time sweeps of the automata calculations as $\bar{x}$ passes through them.

Through A_1:

11	10	9	8	7	6	5	4	3	2	1	0	t
	1	3	0	0	1	2	2	2	0	3	1	x_t
1	0	0	0	1	1	0	1	0	0	1	0	q_t
	4	4	0	3	2	1	2	1	0	3	4	y_t

(5)

Through A_2:

11	10	9	8	7	6	5	4	3	2	1	0	t
	1	3	0	0	1	2	2	2	0	3	1	x_t
1	0	0	1	0	1	0	1	0	1	1	0	q_t
	4	4	0	3	2	1	2	1	0	3	4	y_t

(6)

The symbol t denotes the automaton time and simultaneously for rows x_t and y_t the digit number; the initial state of the automata in both cases is $q = 0$.

The results of automata are read in the lower rows of sweeps (5) and (6), in both cases they coincide and are equal to the object $\bar{y}$, as they should be.

The fact that the algorithm singled out two automata A_1 and A_2 in the class ${}_0\mathcal{A}_0$ means that the information contained in the mapping fragment $\bar{x} \to \bar{y}$ is insufficient for the unambiguous selection of the automaton. To uniquely find an automaton in the class ${}_0\mathcal{A}_0$, we add a mapping $\bar{x} \to \bar{y}$ in the 11th digit and pass to a new mapping:

11	10	9	8	7	6	5	4	3	2	1	0	t
0	1	3	0	0	1	2	2	2	0	3	1	x_t
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
3	4	4	0	3	2	1	2	1	0	3	4	y_t

(7)

Let us pass the object $\bar{x}_1 = 01300122203\underset{0}{1}$ through the automata A_1 and A2 and see what we get on their outputs. When passing $\bar{x}_1$ through A_1, the sweep (5) will be completed only by the 12th step:

12	11	10	...	...	1	0	t
	0	1			3	1	x_t
0	1	0			1	0	q_t
	3	4			3	4	y_t

(8)

As can be seen, in step 11, in full accordance with (7), the result is $y = 3$, i.e., the automaton A_1 implements the mapping $\bar{x}_1 \to \bar{y}_1$, where $\bar{y}_1 = 34403212103\underset{0}{4}$.

As $\bar{x}_1$ passes through A_2 we will have the same thing:

12	11	10	...	...	1	0	t
	0	1			3	1	x_t
0	1	0			1	0	q_t
	0	4			4	4	y_t

(9)

But this time for $t = 11$ the automaton A_2 generates a signal $y'_{11} = 0$ different from the required number 3:

$$\bar{x}_1 \xrightarrow{A_2} \bar{y}'_1 \neq \bar{y}_1 \tag{10}$$

The chart (10) states that automaton A_2 does not implement the mapping $\bar{x}_1 \to \bar{y}_1$ and thus there is only one automaton, A_1 (Fig. 4), which implements the mapping $\bar{x}_1 \to \bar{y}_1$, and thus, assuming that the mapping $\bar{x}_1 \to \bar{y}_1$ is representative, there is only one automaton mapping f1, for which $\bar{x}_1 \to \bar{y}_1$. Thus by a single mapping one can find the whole automaton mapping.

If at t=11 we required, say, a mapping of digits: $\begin{matrix}0\\\downarrow\\1\end{matrix}$, or $\begin{matrix}0\\\downarrow\\2\end{matrix}$, or $\begin{matrix}0\\\downarrow\\4\end{matrix}$ (as well as some more), we would not find any automaton implementing a mapping $\bar{x} \to \bar{y}$ with the specified mappings in the 11th bit. In that case, the algorithm would go on to the search of the ACDs in the classes ${}_1\mathcal{A}_0$, ${}_0\mathcal{A}_1$, etc.

The fact of constructing the automata A_1 by mapping $\bar{x}_1 \to \bar{y}_1$ can be interpreted as a learning process, with which the synthesis of A_1 by mapping $\bar{x}_1 \to \bar{y}_1$ bears a great resemblance.

Indeed, in the brain of a schoolboy when teaching him, say, the operation of addition, after he has performed a finite number of examples, something is formed that allows him to further solve any example on the performance of the operation of addition.

Here we have an analogy. The external constructor, which contains the algorithm for synthesizing from fragments of mappings, passes a *finite* number of correspondences (in the example we have considered one: $\bar{x}_1 \to \bar{y}_1$), and as a result forms a structure (automaton A_1), which allows finding *any* other correspondence of automaton mapping f_1.

Of course, we are not claiming that the same thing happens in the brain; we are merely emphasizing an analogy. Moreover, we will not seek to model nature, not only out of ignorance of how it does it but also because the quantitative effect in achieving the same goal is most often achieved by paths different from those followed by nature.

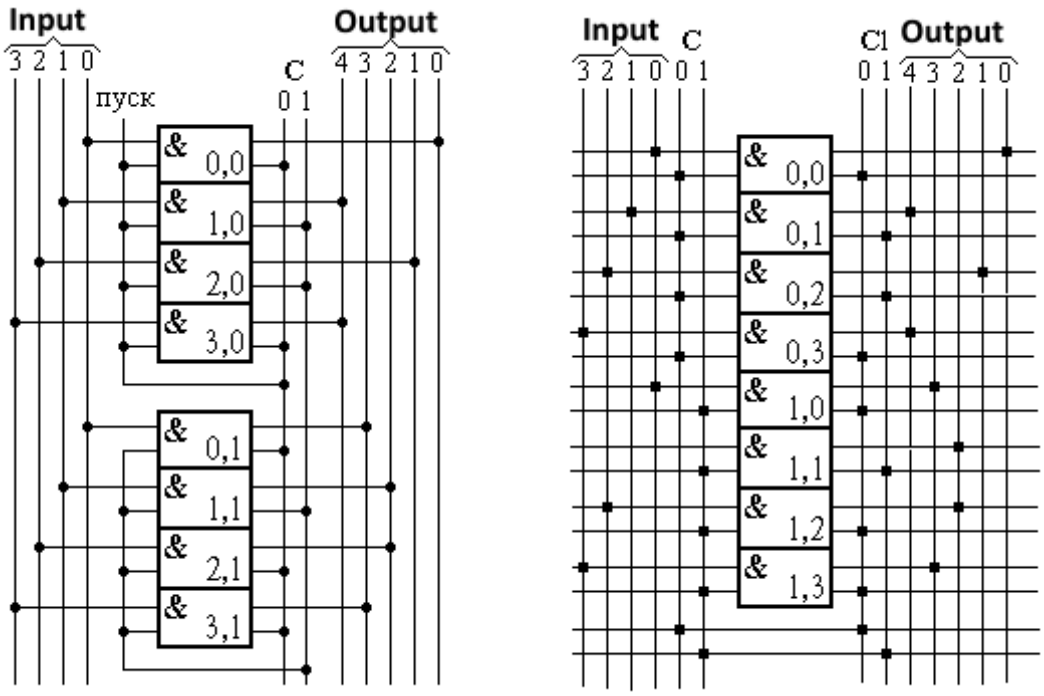

Figure 4. *The canonical B-scheme of automata A_1 (left) and its duplicate (right), made on the template*

A total of 2^{234} various connection options are possible on this type of template. One of them is shown.

The effect of reducing the role of programs in B-computers

Consider a typical fragment of a program code executed on an A-computer (see Fig. 5). The program contains blocks in which functions are calculated. The calculation is performed programmatically. After completing calculations in a block, control is transferred to the next block, in which the same happens. Note that each block calculation takes place programmatically.

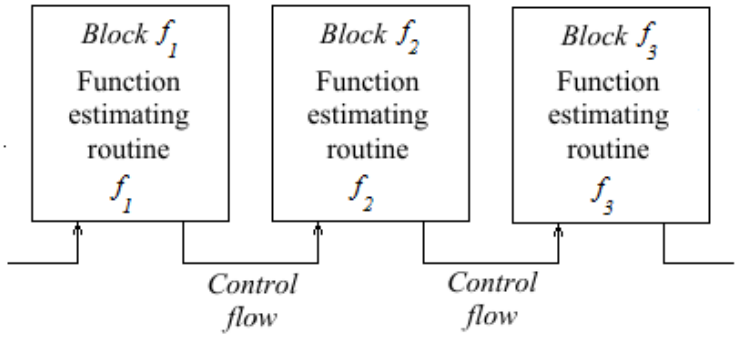

Figure 5. *The linear part of the A-computer program*

Let us compare this situation with a similar one in the B-computer. Let's picture the same part of the program, but done by a B-computer. The difference is that calculation in blocks is not done programmatically, but with the help of devices.

Let us immediately note that this part of the program will be executed by the B-computer faster

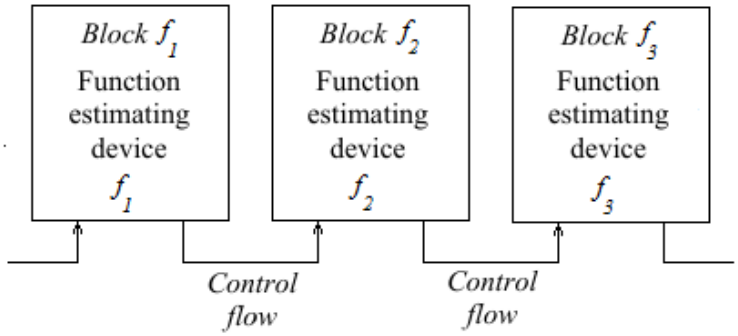

Figure 6. *The same linear part of the program, but executed by the B-computer*

than by the A-computer since the computation of the function by the device is faster than the program computation. But let us not only look at this fact but also at the fact that a large part of the program component has disappeared. This is exactly what we have called the diminishing role of programs in B-computers. In the long term, we can talk about almost completely leveling out the role of programs in B-computers. And what will remain? There will remain the part of programs which is connected with management functions, which is marked in the figures as "management transfer". But this, too, can be implemented in the future by hardware, using devices.

Conclusion

The above brief analysis quite convincingly shows that B-computers can increase their "iron mass" unlimitedly, but not at the expense of single-type boxes piled up within a certain volume, but at the expense of synthesis of various computing devices and their subsequent integration into the computer. The variety of computing devices integrated into the B-computer indicates the "sophistication" of the B-computer, its ability to solve various and complex computational problems. These examples show that synthesis of computing devices embedded into B-computer can be performed by the computer itself inside itself, as it comes down to putting points on templates of blanks – the simplest of all possible operations. The solution to the problem of creating artificial intelligence can be obtained by using B-computers since it is B-computers that make it possible to get rid of the influence of the human factor, which manifests itself through programs, whose influence in B-computers tends to level out.

REFERENCES

1. Deev G. E., Ermakov S. V., Korol N. A., Pereguda A. I., Starkov S. O. Computers for k-based Number System. *Supercomputing and Simulation*. XVI International Conference. October 3–7, 2016, Sarov. (In Russ.)
2. Deev G. E., Lamkov A. V. *Abstract Computers.* In 2 volumes. V. 1. M.: Energoatomizdat; 2004. 548 p. (In Russ.)
3. Deev G. E. *Abstract Computers.* In 2 volumes. V. 2. M.: Energoatomizdat; 2007. 332 p. (In Russ.)
4. Deev G. E. *The Theory of Computers.* St. Petersburg: Lan'; 2019. 452 p. (In Russ.)
5. Minskii M. *Computation: Finite and Infinite Machines*. M.: Mir; 1971. 362 p. (In Russ.)
6. Rich E. *Artificial Intelligence*. New York: McGraw-Hill; 1983. 411 p.
7. Andrew A. M. *Artificial Intelligence, Viable Systems Chillaton*. Devon (U. K.): Abacus Press; 1983.
8. Feigenbaum E. A., Feldman J. *Computers and Thought.* AAAI Press, 1995.
9. Turing A. *Can Machines Think?* Moscow: PhysMathGiz; 1960. 112 p. (In Russ.)
10. Raphael B. *The Thinking Computer: Mind Inside Matter*. W. H. Freeman and Company; 1976. 322 p.
11. Dreyfus H. L. *What Computers Can't Do: A Critique of Artificial Reason*. 1st edition. Harper & Row; 1972. 259 p.
12. Timofeev A. V. *Robots and Artificial Intelligence*. M.: Nauka; 1978. 192 p. (In Russ.)