DOI 10.1007/s10559-019-00154-3
Cybernetics and Systems Analysis, Vol. 55, No. 3, May, 2019

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

PARTITIONING A SET OF VECTORS WITH INTEGER COORDINATES BY MEANS OF LOGICAL HARDWARE

S. L. Kryvyi,[1] V. M. Opanasenko,[2] and S. B. Zavyalov[3]

UDC 516.813

Abstract. *The problem of partitioning a set of vectors with integer coordinates is considered with respect to the coordinate-wise and lexicographic order on vectors using automaton interpretation. An FPGA-based hardware implementation of three-valued logic operations is proposed to check the satisfiability of formulas of this logic.*

Keywords: *integer-valued vector, threshold value, finite automaton, three-valued logic.*

INTRODUCTION

Synthesizing hardware to solve the problem of partitioning a set of Boolean vectors and vectors with integer-valued non-negative coordinates with respect to a threshold value was considered in [1] (using algorithms from [2, 3]). This approach is well-known as "the technology of reconfigurable computing" [4, 5] and its implementation in real projects (see [6–13]) became possible owing to the emergence of the programmable logic integrated circuits (PLIC).

This article considers a generalization of the method for solving the problem of partitioning a set of vectors, which was considered in [1], and the use of this method to check the satisfiability of formulas of three-valued logic.

PROBLEM STATEMENT

A finite set of vectors $V = \{v_1, v_2, \dots, v_m\}$ of dimension $n \in N$ and a fixed threshold vector $a = (c_1, c_2, \dots, c_n)$ are given, where v_i, $a \in Z^n$, and Z is the set of integer numbers. An order relation is specified on the set of vectors. Most frequently, such a relation is the coordinate-wise (partial) order or lexicographic (complete) order.

Definition 1. Let $x = (x_1, \dots, x_n) \in Z^n$, and let $y = (y_1, \dots, y_n) \in Z^n$. The binary relation $x \le y \Leftrightarrow \forall i = 1, \dots, n$, $x_i \le y_i$ is called the coordinate-wise order.

The binary relation $x < y \Leftrightarrow (\forall i < j\ x_i = y_i) \wedge (x_j < y_j)$, $i, j = 1, \dots, n$, is called the lexicographic order.

We consider that, with respect to the coordinate-wise order $x < y$, for at least one $i = 1, \dots, n$, we have $x_i < y_i$.

Partition problem. It is necessary to partition the set V with respect to the threshold vector a into the subsets $V_1 = \{v \in V \mid v < a\}$, $V_2 = \{v \in V \mid v > a\}$, $V_3 = \{v \in V \mid v = a\}$, and $V_4 = \{v \in V \mid v \propto a\}$, where the symbol $\propto$ means that the corresponding vectors are not comparable (there are the following three subsets for the lexicographic order: V_1, V_2, and V_3).

This article considers the coordinate-wise order since the solution of the partition problem for the lexicographic order obviously follows from the solution of the partition problem for the coordinate-wise order.

[1]Taras Shevchenko National University of Kyiv, Kyiv, Ukraine, *sl.krivoi@gmail.com*. [2]V. M. Glushkov Institute of Cybernetics. National Academy of Sciences of Ukraine, Kyiv, Ukraine, *OpanasenkoVM@nas.gov.ua*. [3]LLC "Radioniks," Kyiv, Ukraine, *radionix13@gmail.com*. Translated from Kibernetika i Sistemnyi Analiz, No. 3, May–June, 2019, pp. 136–148. Original article submitted September 12, 2018.

 1060-0396/19/5503-0462

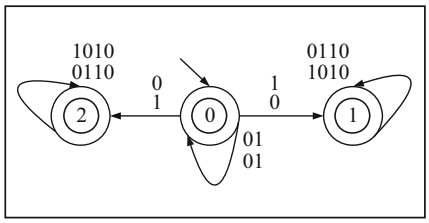

Fig. 1. Automaton A_1.

AUTOMATON INTERPRETATION OF THE PARTITION PROBLEM

Case of integer nonnegative coordinates of vectors. The automaton approach to the solution of the partition problem consists of constructing a homogeneous network from simple automata whose final states determine the membership of a vector $v \in V$ in one of the subsets V_1, V_2, V_3, or V_4. An advantage of the automaton approach is that it allows partitioning the set V into the subsets V_1, V_2, V_3, and V_4 simultaneously for all the relations $R \in \{<, >, =, \propto\}$. The automata from which the network is constructed are automata without outputs [14, 15], and the formal definition of a network of automata is as follows.

Definition 2. A network of automata is understood to be an n-tuple $\boldsymbol{A} = (A_1, \ldots, A_n)$ consisting of automata presented in the form $A_i = (S_i, X_i, f_i, a_0^i, F_i)$, $i = 1, 2, \ldots, n$. A state of the network $\boldsymbol{A}$ is an n-tuple of states $(a_1, \ldots, a_n)$, where $a_i \in S_i$ for each $i = 1, 2, \ldots, n$. A network state $(a_1, \ldots, a_n)$ is called initial if $a_i = a_0^i$ and final if $a_i \in F_i$ for all $i = 1, 2, \ldots, n$. Then an n-tuple $x = (x_1, \ldots, x_n)$, where $x_i \in X_i$, $i = 1, 2, \ldots, n$, is called an action in the network $\boldsymbol{A}$ that switches the network from the state $(a_1, \ldots, a_n)$ into a state $(b_1, \ldots, b_n)$ so that $(b_1, \ldots, b_n) = (f_1(a_1, x_1), \ldots, f_n(a_n, x_n))$.

A network is called homogeneous if all its automata are of the same form.

To solve the classification problem, a homogeneous network of automata [14, 15] is used whose form is shown in Fig. 1. The initial state of an automaton is its zero state, all the states of an automaton are final, and its input alphabet consists of column vectors in binary alphabet that represent the corresponding pairs of coordinates of a vector v from V and the threshold vector a.

We will explain this representation by an example.

Example 1. Let $v = (3, 4)$, and let $a = (3, 2)$. These vectors are represented by the binary words $v^{\mathrm{T}} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} = \begin{bmatrix} 011 \\ 100 \end{bmatrix} = v_3 v_2 v_1$, where $v_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $v_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $v_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $a^{\mathrm{T}} = \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{bmatrix} 011 \\ 010 \end{bmatrix} = a_3 a_2 a_1$, where $a_3 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $a_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $a_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. Thus, the alphabet of the automaton A_1 consists of the symbols $x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$, $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $x_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, and $x_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$. The following words are applied to the input of the automaton A_1: $p_1 = \begin{bmatrix} 011 \\ 011 \end{bmatrix} = x_1 x_4 x_4$ and $p_2 = \begin{bmatrix} 100 \\ 010 \end{bmatrix} = x_3 x_2 x_1$ beginning with their high-order digits; these words represent the first and second coordinates of the vectors v and a, respectively.

End of Example 1.

The set to which belongs the vector v is determined depending on the final state at which the network of automata $\boldsymbol{A}$ completes its operation. In the above example, we have the vector $v > a$ with respect to the coordinate-wise and lexicographic orders, and it should be attributed to the subset V_2.

Proposition 1. The automaton A_1 correctly computes the relation $x \le y$ between two integer positive numbers x and y represented in binary number system.

The proof obviously follows from the fact that the order relation $x \le y$ on vectors is extended to digits of their binary representation of coordinates. Indeed, let $p_x = x_1 x_2 \dots x_k$ and $p_y = y_1 y_2 \dots y_k$ be binary words representing the numbers x and y, respectively. Since binary words are applied to the input of the automaton A_1 beginning with their high-order digits, $x < y$ if $x_j < y_j$ for some $1 \le j \le k$ and $x_i = y_i$ for all $i < j$. This means that the jth digit of the number x is less than the jth digit of the number y, and then the automaton transits from state 0 to state 2 (see Fig. 1). This means that $x < y$. The other cases are similar.

End of the proof.

A general solution of the problem of partitioning the set V is find by the network of automata $\boldsymbol{A} = (A_1, \dots, A_n)$ (the case of n-dimensional vectors from V). Tuning a network to solve the partition problem is as follows: all the automata of the network are identical (n instances of the automaton A_1 in Fig. 1); the network is in its initial state; words are applied to the inputs of the automata of the network beginning with high-order digits. The binary word representing the first coordinates of the vectors v and a is applied to the input of the first automaton of the network, the word representing the second coordinates of the vectors is applied to the input of the second automaton of the network, etc. As has been noted above, the membership of the vector v in a subset V_i depends on the final network state to which the automaton transits after reading input words. If the order relation is coordinate-wise and the final state of the network is of the form

(a) (0, 0, 0), then $v = a$ and, therefore, $v \in V_3$;

(b) (p, q, r), where $p, q, r \in \{0, 1\}$ and not all the values of p, q, and r are equal to zero, then $v > a$ and, therefore, $v \in V_2$;

(c) (p, q, r), where $p, q, r \in \{0, 2\}$ and not all the values of p, q, and r are equal to zero, then $v < a$ and, therefore, $v \in V_1$;

(d) (1, 2, 0), (0, 1, 2), (2, 1, 0), (2, 1, 2). (2, 2, 1), …, then the vector v is not comparable with a and, therefore, $v \in V_4$.

If the order relation is lexicographic and the final state of the network is

(a′) (0, 0, 0), then $v = a$ and, therefore, $v \in V_3$;

(b′) (p, q, r) and $p < q$ or $p = q$ and $q < r$, then $v > a$ and, therefore, $v \in V_2$;

(c′) (p, q, r) and $p > q$ or $p = q$ and $q > r$, then $v < a$ and, therefore, $v \in V_1$.

Example 2. Assume that $V = \{v_1 = (4, 5, 0), v_2 = (3, 4, 2), v_3 = (0, 1, 1), v_4 = (1, 1, 2),$ and $v_5 = (4, 3, 1)\}$ and that the threshold vector $a = (1, 1, 2)$. Then the pair (v_1, a) is represented as follows:

$$v_1 = \begin{pmatrix} 4 \\ 5 \\ 0 \end{pmatrix} = \begin{bmatrix} 100 \\ 101 \\ 000 \end{bmatrix}, \; a = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} = \begin{bmatrix} 001 \\ 001 \\ 010 \end{bmatrix}$$

and the words corresponding to the first coordinates are of the form $p_1 = \begin{bmatrix} 100 \\ 001 \end{bmatrix}$, those corresponding to the second coordinates are of the form $p_2 = \begin{bmatrix} 101 \\ 001 \end{bmatrix}$, and those corresponding to the third coordinates are of the form $p_3 = \begin{bmatrix} 000 \\ 010 \end{bmatrix}$.

The word p_1 is applied to the input of the first automaton of the network, the word p_2 is applied to the input of the second automaton of the network, and p_3 is applied to the input of the third automaton. As a result, we obtain the states (1, 1, 2) in which the automata have stopped. This means that the vectors v_1 and a are not comparable with respect to the coordinate-wise order, and, with respect to the lexicographic order, $v_1 > a$. Next, for the coordinate-wise order, we obtain the coordinates

$$v_2 = \begin{bmatrix} 011 \\ 100 \\ 010 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 011 \\ 001 \end{bmatrix}, \; p_2 = \begin{bmatrix} 100 \\ 001 \end{bmatrix}, \; p_3 = \begin{bmatrix} 010 \\ 010 \end{bmatrix}$$

that switch the network to the states (1, 1, 0), which signifies that $v_2 > a$;

the coordinates

$$v_3 = \begin{bmatrix} 000 \\ 001 \\ 001 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 000 \\ 001 \end{bmatrix},\ p_2 = \begin{bmatrix} 001 \\ 001 \end{bmatrix},\ p_3 = \begin{bmatrix} 001 \\ 010 \end{bmatrix}$$

that switch the network to the states (2, 0, 2), which signifies that $v_3 < a$;

the coordinates

$$v_4 = \begin{bmatrix} 100 \\ 100 \\ 010 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 001 \\ 001 \end{bmatrix},\ p_2 = \begin{bmatrix} 001 \\ 001 \end{bmatrix},\ \text{and}\ p_3 = \begin{bmatrix} 010 \\ 010 \end{bmatrix}$$

that switch the network into the state (0, 0, 0), which signifies that $v_4 = a$;

the coordinates

$$v_5 = \begin{bmatrix} 100 \\ 011 \\ 001 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 100 \\ 001 \end{bmatrix},\ p_2 = \begin{bmatrix} 011 \\ 001 \end{bmatrix},\ p_3 = \begin{bmatrix} 001 \\ 010 \end{bmatrix}$$

that switch the network to the state (1, 1, 2), which means that the vector v_5 is not comparable with a.

Thus, $V = V_1 \cup V_2 \cup V_3 \cup V_4$, where $V_1 = \{v_3\}, V_2 = \{v_2\}, V_3 = \{v_4\}$, and $V_4 = \{v_1, v_5\}$ is the partition of the set A with respect to the coordinate-wise order.

End of Example 2.

Proposition 2. The homogeneous network of automata $\boldsymbol{A} = (A_1, \ldots, A_n)$ correctly computes the relation $x \le y$ between two vectors x and y with integer positive coordinates that are represented in binary number system.

The proof follows directly from Proposition 1 and items (a)–(d) for the coordinate-wise order relation and from items (a′)–(c′) for the lexicographic order relation. Indeed, let $x < y$; then, according to Proposition 1, one of automata of the network $\boldsymbol{A}$ transits to terminal state 2 (see Fig. 1) and remains in it until the end of operation of the network; the other automata remain in state 0 (equality of coordinates) or transit to state 2 and remain in it until the end of operation of the network. According to item (c), for the coordinate-wise order, $v < a$. For the lexicographic order relation, the proof follows from item (c′). The other cases are similar.

End of the proof.

Case of integer-valued coordinates of vectors. Consider the above problem statement but in which the coordinates of vectors from V are not natural but integer numbers. In this case, the problem will be solved by the method considered above if negative coordinates of vectors are represented in the form of 2's complements. Note that the 2's complement of a number x whose binary representation is $x_n x_{n-1} \ldots x_1 x_0$ is defined as follows:

if $x \ge 0$, then $x_n = 0$ and $(x)_2 = 0 x_{n-1} \ldots x_1 x_0$,

if $x < 0$, then $x_n = 1$ and $(x)_2 = 1\,\overline{x_{n-1}} \ldots \overline{x_0} + 1$, where the bit x_n is the sign bit and $\overline{x_i} = 1 - x_i$ for $0 \le i \le n-1$.

The automaton for computing a 2's complement is represented by the composition of two automata, the first of which constructs the complement and the second implements the operation of adding 1. The composition of these automata is represented by the automaton C shown in Fig. 2.

In Fig. 3, the automaton B_1 is presented; a homogeneous network $\boldsymbol{B}$ is constructed from such automata. The state a_0 is introduced to recognize the signs of the binary numbers representing the coordinates of vectors from V. All the states of the automaton B_1, except for the state a_0 are final. If sign bits are different, then one can immediately determine the largest vector. If sign bits are identical, then positive coordinates are compared as is shown above, and if these bits are negative, then they are compared by the part of the automaton that corresponds to state 3 in Fig. 3.

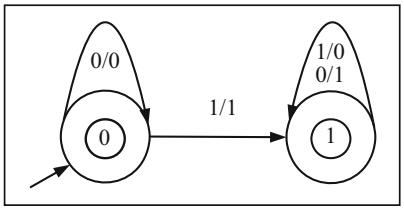

Fig. 2. Automaton C for computing a 2's complement.

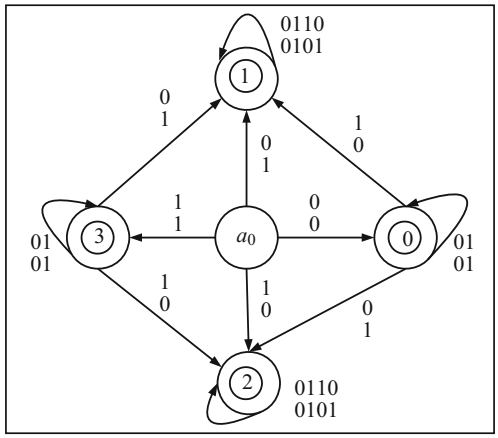

Fig. 3. Automaton B_1.

Example 3. Assume that $V = \{v_1 = (-4, 5, 0),\ v_2 = (3, 4, -2),\ v_3 = (0, -1, 1),$ and $v_4 = (1, 1, 2)\}$ and that the threshold vector $a = (-1, 1, 2)$. Then the pair (v_1, a) is represented as follows:

$$v_1 = \begin{pmatrix} -4 \\ 5 \\ 0 \end{pmatrix} = \begin{bmatrix} 1100 \\ 0101 \\ 0000 \end{bmatrix},\ a = \begin{pmatrix} -1 \\ 1 \\ 2 \end{pmatrix} = \begin{bmatrix} 1111 \\ 0001 \\ 0010 \end{bmatrix},$$

words corresponding to the coordinates are

$$p_1 = \begin{bmatrix} 1100 \\ 1111 \end{bmatrix},\ p_2 = \begin{bmatrix} 0101 \\ 0001 \end{bmatrix},\ p_3 = \begin{bmatrix} 0000 \\ 0010 \end{bmatrix},$$

and the result is represented by the state (1, 1, 2). This means that the vectors v_1 and a are not comparable.

For v_2 and a, we have

$$v_2 = \begin{bmatrix} 0011 \\ 0100 \\ 1110 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 1100 \\ 1111 \end{bmatrix},\ p_2 = \begin{bmatrix} 0100 \\ 0001 \end{bmatrix},\ p_3 = \begin{bmatrix} 1110 \\ 0010 \end{bmatrix},$$

and the result is the state (1, 1, 2); consequently, v_2 and a are also not comparable.

For v_3 and a, we have

$$v_3 = \begin{bmatrix} 0000 \\ 1111 \\ 0001 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 0000 \\ 1111 \end{bmatrix},\ p_2 = \begin{bmatrix} 1111 \\ 0001 \end{bmatrix},\ p_3 = \begin{bmatrix} 0001 \\ 0010 \end{bmatrix},$$

and the result is the state (1, 2, 1); this means that v_3 and a are not comparable.

For v_4 and a, we have

$$v_4 = \begin{bmatrix} 0001 \\ 0001 \\ 0010 \end{bmatrix} \Rightarrow p_1 = \begin{bmatrix} 0001 \\ 1111 \end{bmatrix},\ p_2 = \begin{bmatrix} 0001 \\ 0001 \end{bmatrix},\ p_3 = \begin{bmatrix} 0010 \\ 0010 \end{bmatrix},$$

and the result is (1, 0, 0), which means that $a < v_4$.

End of Example 3.

TABLE 1. Results of Comparison of Vectors A and B Represented in Direct and Complementary Codes

Direct Binary Code			**Complementary Binary Code**		
Binary vector A_j	Binary vector B_i	Result of comparison	Binary vector A_j	Binary vector B_i	Result of comparison
$A_1 = 000$ (+0)	$B_1 = 000$ (+0)	$A = B$ (Equal)	$A_1 = 000$ (+0)	$B_1 = 000$ (+0)	$A = B$ (Equal)
	$B_2 = 010$ (+2)	$B > A$ (More)		$B_2 = 010$ (+2)	$B > A$ (More)
	$B_3 = 111$ (–3)	$B < A$ (Less)		$B_3 = 101$ (–3)	$B < A$ (Less)
	$B_4 = 110$ (–2)	$B < A$ (Less)		$B_4 = 110$ (–2)	$B < A$ (Less)
	$B_5 = 101$ (–1)	$B < A$ (Less)		$B_5 = 111$ (–1)	$B < A$ (Less)
$A_2 = 001$ (+1)	$B_1 = 000$ (+0)	$B < A$ (Less)	$A_2 = 001$ (+1)	$B_1 = 000$ (+0)	$B < A$ (Less)
	$B_2 = 010$ (+2)	$B > A$ (More)		$B_2 = 010$ (+2)	$B > A$ (More)
	$B_3 = 111$ (–3)	$B < A$ (Less)		$B_3 = 101$ (–3)	$B < A$ (Less)
	$B_4 = 110$ (–2)	$B < A$ (Less)		$B_4 = 110$ (–2)	$B < A$ (Less)
	$B_5 = 101$ (–1)	$B < A$ (Less)		$B_5 = 111$ (–1)	$B < A$ (Less)
$A_3 = 011$ (+3)	$B_1 = 000$ (+0)	$B < A$ (Less)	$A_3 = 011$ (+3)	$B_1 = 000$ (+0)	$B < A$ (Less)
	$B_2 = 010$ (+2)	$B < A$ (Less)		$B_2 = 010$ (+2)	$B < A$ (Less)
	$B_3 = 111$ (–3)	$B < A$ (Less)		$B_3 = 101$ (–3)	$B < A$ (Less)
	$B_4 = 110$ (–2)	$B < A$ (Less)		$B_4 = 110$ (–2)	$B < A$ (Less)
	$B_5 = 101$ (–1)	$B < A$ (Less)		$B_5 = 111$ (–1)	$B < A$ (Less)
$A_4 = 110$ (–2)	$B_1 = 000$ (+0)	$B > A$ (More)	$A_4 = 110$ (–2)	$B_1 = 000$ (+0)	$B > A$ (More)
	$B_2 = 010$ (+2)	$B > A$ (More)		$B_2 = 010$ (+2)	$B > A$ (More)
	$B_3 = 111$ (–3)	$B < A$ (Less)		$B_3 = 101$ (–3)	$B < A$ (Less)
	$B_4 = 110$ (–2)	$A = B$ (Equal)		$B_4 = 110$ (–2)	$A = B$ (Equal)
	$B_5 = 101$ (–1)	$B > A$ (More)		$B_5 = 111$ (–1)	$B > A$ (More)

Proposition 3. The automaton B_1 correctly computes the relation $x \le y$ between two integer numbers x and y that are represented in binary number system in the form of their 2's complements.

The proof obviously follows from Proposition 1. Indeed, when transiting from the initial state a_0, the automaton B_1 recognizes the sign, passes to the state 0 if the numbers are positive, and them operates as the automaton A_1. If the numbers are negative, then the automaton B_1 passes to the state 3 and then operates as the automaton A_1, only the transitions from the state 3 to the state 1 or 2 are asymmetric to the transitions of the automaton A_1 from the state 0 to the state 1 or 2. If the signs are different, then the automaton B_1 immediately passes either to the state 1 (the case when $x > y$) or to the state 2 (the case when $x < y$). In the process of comparison, passing to the state 1 (or 2), the automaton B_1 remains in it. This means that the comparison is performed unambiguously.

End of the proof.

The comparison of vectors of length n is performed by a homogeneous network composed of automata B_1.

THEOREM 1. The networks ***A*** and ***B*** correctly partition a set of vectors with respect to the coordinate-wise and lexicographic orders and a threshold vector.

The proof follows from Propositions 1–3.

The correctness of functioning the homogeneous network ***B*** for the lexicographic order is also illustrated by the results of testing given in Table 1.

IMPLEMENTATION OF COMPARING TWO VECTORS BASED ON LOGICAL STRUCTURES

Since the classification problem is solved by a homogeneous network of automata, it suffices, based on a logical structure, to implement only the operations of comparing of two vectors, i.e., only the automata A_1 and B_1. For an arbitrary vector $v \in V$ of dimension $n \in N$ and a fixed threshold vector a, where $v, a \in Z^n$ and Z is the set of integer numbers, it is necessary to form the following results of comparison: $v < a$ (Less), $v > a$ (More), and $v = a$ (Equal).

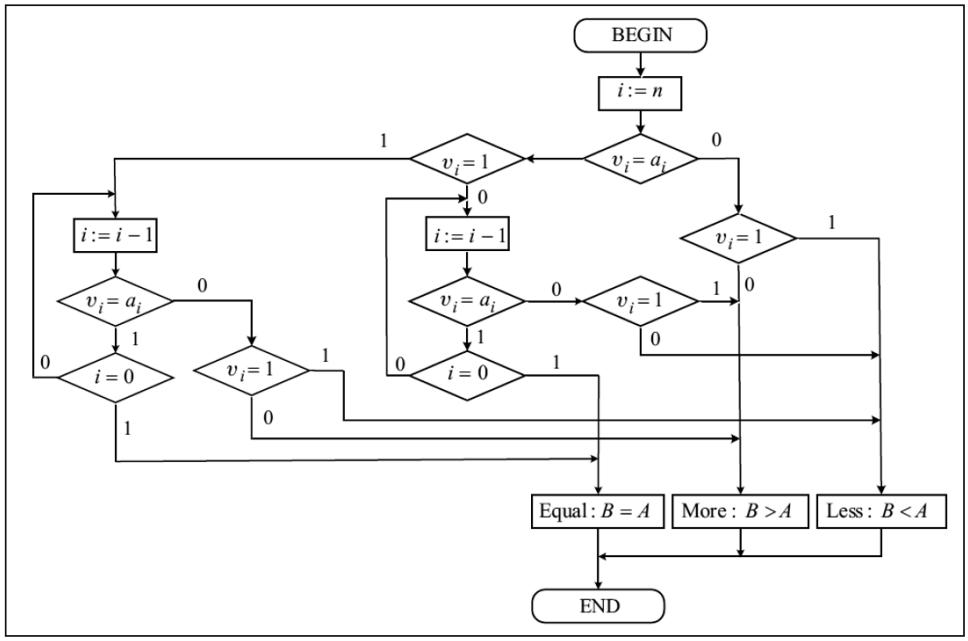

Fig. 4. Block diagram of algorithm for comparing signed binary numbers.

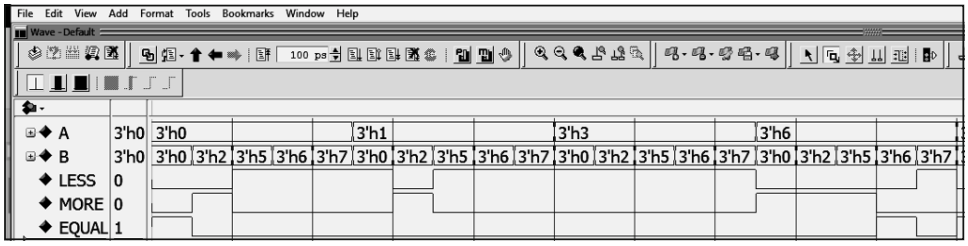

Fig. 5. Time diagram of the algorithm for comparing binary signed numbers in direct code.

Let A and B ($A = a$ and $B = v$) be binary vectors of dimension n that are represented as follows: $A = a_n a_{n-1} a_{n-2} \ldots a_0$ and $B = b_n b_{n-1} b_{n-2} \ldots b_0$, where a_n and b_n are sign bits of A and B. The block diagram of an algorithm for comparing binary vectors (in direct codes) is presented in Fig. 4. It demonstrates the result of comparison of two vectors.

Example 4. Let A and B be sets of vectors with binary representation of their coordinates, i.e., $a = a_n a_{n-1} a_{n-2} \ldots a_0 \in A$ and $b = b_n b_{n-1} b_{n-2} \ldots b_0 \in B$, where a_n and b_n are the sign bits of a and b. Assume that, for the case when $n = 3$, we have

$$A = \{A_1 = 000;\ A_2 = 001;\ A_3 = 011;\ A_4 = 110\},$$

$$B = \{B_1 = 000;\ B_2 = 010;\ B_3 = 111;\ B_4 = 110;\ B_5 = 101\}.$$

It is required to find the partition of vectors from the set B with respect to the lexicographic order and corresponding threshold vectors from the set A for a relation $R \in \{<, >, =\}$.

We now consider the implementation of a comparison algorithm (numbers are represented in direct codes) based on FPGA crystals using a CAD system with subsequent modeling in the environment of ModelSim. The results of modeling (the time diagram presented in Fig. 5) confirm the correctness of functioning the structure for comparing signed numbers and have the following designations:

$$A_1 = 000 = 3'h0,\ A_2 = 001 = 3'h1,\ A_3 = 011 = 3'h3,\ A_4 = 110 = 3'h6;$$

$$B_1 = 000 = 3'h0,\ B_2 = 010 = 3'h2,\ B_3 = 111 = 3'h7,\ B_4 = 110 = 3'h6,\ B_5 = 101 = 3'h5.$$

The considered example of implementation of the algorithm for comparing two-digit signed numbers according to the results of modeling is executed in an FPGA crystal during several nanoseconds. With increasing the digit capacity of numbers, the time of executing the operation is not practically changed since the implementation presumes parallel digit-by-digit comparison.

End of an Example 4.

IMPLEMENTATION OF OPERATIONS IN THREE-VALUED LOGIC

Consider a method for checking the satisfiability of formulas of Lukasiewicz's three-valued logic [16] that is also applicable to Kleene's three-valued logic, to Bochvar's three-valued logic, and also to any k-valued logic. The possibility of using the described method is based on modeling the operations of three-valued logic using two-valued logic. We will present the table of operations of Lukasiewicz's three-valued logic in which the values 1, 1/2, 0 (Table 2) are used and the table of modeling these operations by two-valued logic (Table 3).

To provide the correspondence with two-valued logic, the values of three-valued logic are coded as follows: $1 \to 10,\ 1/2 \to 01,$ and $0 \to 00$.

Since two bits are used for coding values of three-valued logic, outputs will also be two-bit values. Let $x = x_1 x_0$, let $y = y_1 y_0$, and let $z = z_1 z_0$ be the bit representation of input and output values of variables. We will construct the following algebraic expressions for the presented logical operations with allowance for the adopted encoding:

(1) the operation $A \wedge B$ is specified as follows:

$$z_0 = \overline{x_1} x_0 (y_1 \overline{y_0} + \overline{y_1} y_0) + \overline{x_0} x_1 \overline{y_1} y_0,\ z_1 = \overline{x_0} x_1 y_1 \overline{y_0},$$

the scheme of implementation of this operation is presented in Fig. 6;

(2) the operation $A \vee B$ is specified as follows:

$$z_0 = \overline{x_1}\,\overline{y_1}(x_0 + \overline{x_0} y_0),\ z_1 = \overline{x_1} y_1 \overline{y_0} + \overline{x_0} x_1 (y_1 \overline{y_0} + \overline{y_1}),$$

the scheme of implementation of this operation is presented in Fig. 7;

(3) the operation $A \to B$ is specified as follows:

$$z_0 = \overline{y_1}(x_1 \overline{x_0} y_0 + \overline{x_1} x_0 \overline{y_0}),\ z_1 = y_1 \overline{y_0}(\overline{x_1} + \overline{x_0} x_1) + \overline{x_1}\,\overline{y_1}(\overline{x_0} + x_0 y_0),$$

the scheme of implementation of this operation is presented in Fig. 8;

(4) the operation $\overline{A}$ is specified as follows:

$$z_0 = \overline{x_1} x_0,\ z_1 = \overline{x_0}\,\overline{x_1},$$

the scheme of implementation of this operation is presented in Fig. 9.

We now consider the implementation of operations of three-valued logic (which corresponds to Table 3) based on FPGA crystals using a CAD system with subsequent modeling in the environment ModelSim. The results of modeling (the time diagram presented in Fig. 10) confirm the correctness of functioning the structures of three-valued logic. The following designations are used: Zand, Zor, AsB, and invA correspond to the logical operations $A \wedge B$, $A \vee B$, $A \to B$, and $\overline{A}$, and the states $2'h0$, $2'h1$, and $2'h2$ correspond to the values of three-valued logic in the encoding $0 \to 00,\ 1/2 \to 01,$ and $1 \to 10$; $2'h3$ is the forbidden state.

THEOREM 2. The presented implementation of the logical operations for three-valued logic is correct, i.e., the values of these operations are correctly computed.

The proof is reduced to the consideration of the correctness of implementation of each operation. Consider the proof only for the case of the conjunction and disjunction operations since it is similar for the other operations.

TABLE 2. Truth Values of Logical Functions for Three-Valued Logic

$A = x$	$B = y$	min (x, y) corresponds to $A \wedge B$	max (x, y) corresponds to $A \vee B$	min $(1, 1 - x + y)$ corresponds to $A \to B$	$(1 - x)$ corresponds to $\bar{A}$
1	1	1	1	1	0
1	1/2	1/2	1	1/2	0
1	0	0	1	0	0
1/2	1	1/2	1	1	1/2
1/2	1/2	1/2	1/2	1	1/2
1/2	0	0	1/2	1/2	1/2
0	1	0	1	1	1
0	1/2	0	1/2	1	1
0	0	0	0	1	1

TABLE 3. Modeling Truth Values of Logical Functions for Three-Valued Logic

$A = x$	$B = y$	min (x, y) corresponds to $A \wedge B$	max (x, y) corresponds to $A \vee B$	min $(1, 1 - x + y)$ corresponds to $A \to B$	$(1 - x)$ corresponds to $\bar{A}$
10	10	10	10	10	00
10	01	01	10	01	00
10	00	00	10	00	00
01	10	01	10	10	01
01	01	01	01	10	01
01	00	00	01	01	01
00	10	00	10	10	10
00	01	00	01	10	10
00	00	00	00	10	10

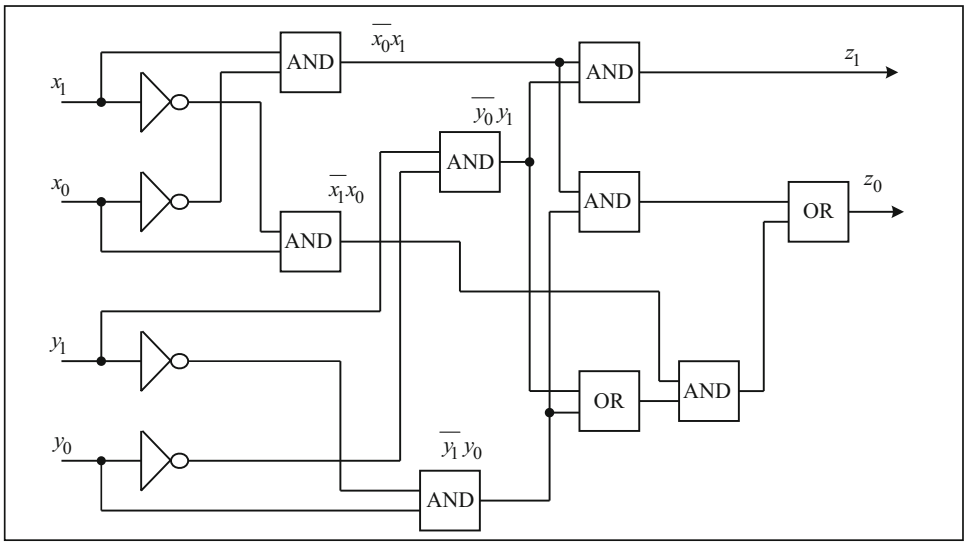

Fig. 6. Scheme of implementation of the operation $A \wedge B$.

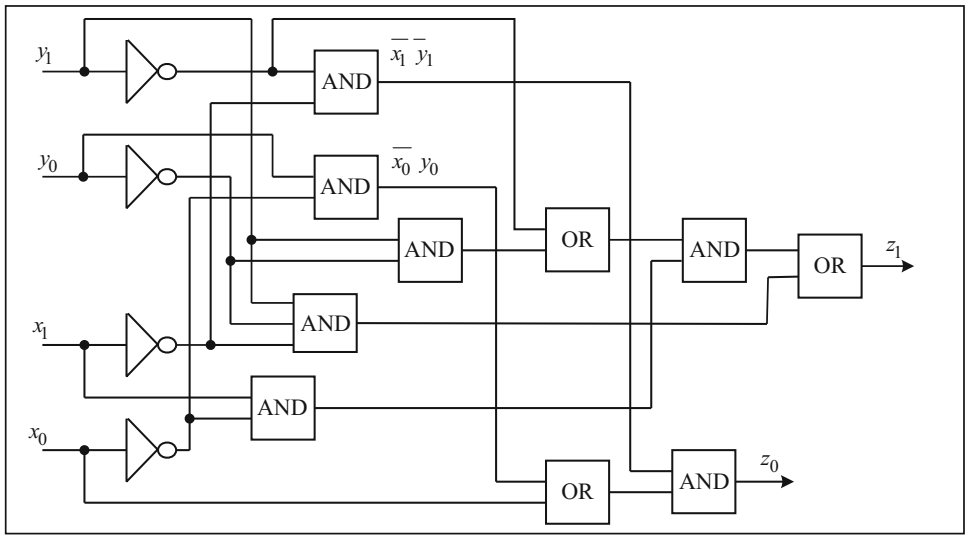

Fig. 7. Scheme of implementation of the operation $A \vee B$.

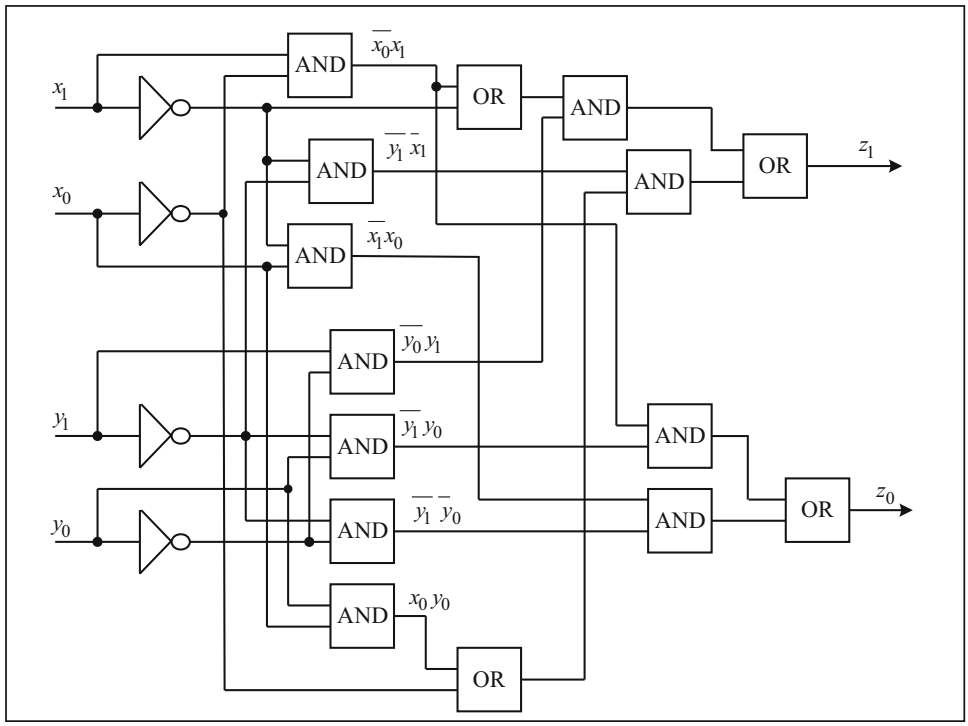

Fig. 8. Scheme of implementation of the operation $A \to B$.

According to the encoding table, to the operation $A \wedge B$ corresponds the DNF $z_0 = \overline{x_0} x_1 \overline{y_1} y_0 + \overline{x_1} x_0 y_1 \overline{y_0} + \overline{x_1} x_0 \overline{y_1} y_0$ that is transformed into the following final expression: $z_0 = \overline{x_1} x_0 (y_1 \overline{y_0} + \overline{y_1} y_0) + x_0 x_1 \overline{y_1} y_0$ and, for $z_1 = \overline{x_0} x_1 y_1 \overline{y_0}$, any transformations are not required. It is these expressions that are implemented by the scheme for $A \wedge B$.

According to the encoding table, to the operation $A \vee B$ corresponds the DNF of the form $z_0 = x_1 x_0 y_0 \overline{y_1} + x_1 x_0 \overline{y_0} y_1 + y_1 y_0 \overline{x_0} \overline{x_1}$ that is transformed into the following final expression: $z_0 = x_1 y_1 (x_0 + x_0 y_0)$ and, for z_1, its DNF is of the form $z_1 = \overline{x_0} x_1 y_1 \overline{y_0} + \overline{x_0} x_1 y_1 \overline{y_0} + \overline{x_0} x_1 \overline{y_0} y_1 + y_1 \overline{y_0} \overline{x_1} x_0 + y_1 \overline{y_0} \overline{x_1} \overline{x_0}$ that is transformed into the following final expression: $z_1 = \overline{x_1} y_1 \overline{y_0} + \overline{x_0} x_1 (y_1 \overline{y_0} + \overline{y_1})$. It is precisely these expressions that are implemented by the scheme for $A \vee B$.

End of the proof.

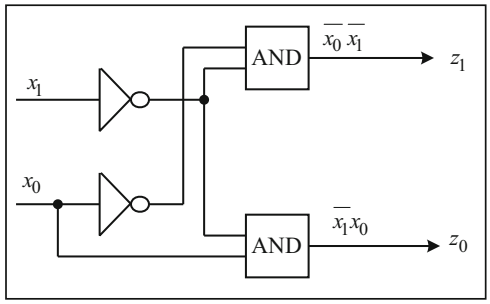

Fig. 9. Scheme of implementation of the operation $\overline{A}$.

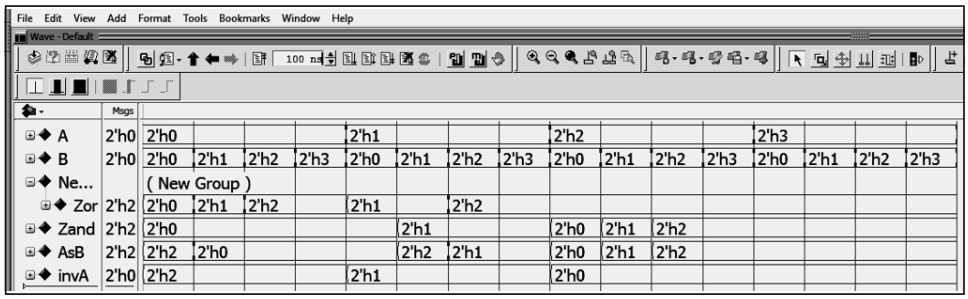

Fig. 10. Time diagram of functioning the operations of three-valued logic.

Example 5. Let us compute the value of the following formula in Lukasiewicz's three-valued logic:

$$F = ((\mathrm{A} \to \mathrm{B}) \vee (\mathrm{C} \wedge \mathrm{D})) \wedge \neg \mathrm{B}.$$

To compute the value of this formula, the network is synthesized based on the structure of the formula F. The synthesized network assumes the form

$$S_1(S_2(S_3(\mathrm{A}, \mathrm{B}), S_1(\mathrm{C}, \mathrm{D})), S_4(\mathrm{B})),$$

where S_1, S_2, S_3, and S_4 are the schemes of implementing the functions of conjunction, disjunction, implication, and negation, respectively. In particular, let the following input values of the variables be given: $\mathrm{A} = 1$, $\mathrm{B} = 0$, $\mathrm{C} = 1/2$, and $\mathrm{D} = 0$; then we obtain the following value of the formula F at the output:

$$S_3(10, 00) = 00, \;\; S_1(01, 00) = 00, \;\; S_4(00) = 10,$$

$$S_1(S_2(S_3(10, 00), S_1(01, 00)), S_4(00)) = S_1(S_2(00, 00), S_4(00)) = S_1(00, 10) = 00.$$

Thus, the value of the formula is equal to zero, which corresponds to the value computed according to Table 2. For the values of $\mathrm{A} = 1$, $\mathrm{B} = 1$, $\mathrm{C} = 1/2$, and $\mathrm{D} = 1/2$, we obtain the following value of the formula F:

$$S_3(10, 10) = 10, \;\; S_1(01, 01) = 01, \;\; S_4(10) = 00,$$

$$S_1(S_2(S_3(10, 10), S_1(01, 01)), S_4(10)) = S_1(S_2(10, 01), S_4(10)) = S_1(10, 00) = 00.$$

End of Example 5.

It follows from Example 5 that this approach allows to synthesize a network from the structure of a specified formula in three-valued logic and, using this network, to check this formula for the satisfiability for the values of the input variables.

CONCLUSIONS

This article considers the solution of the problem of partitioning a set of vectors with integer coordinates with respect to the coordinate-wise and lexicographic orders on vectors. In this case, the automaton interpretation of the partition problem was used that solves the problem in general form. The presented solution is independent of the digit capacity of coordinates of vectors and is applicable to arbitrary values of coordinates of vectors and threshold values. Thus, the methods for solving partition problems are generalized that were considered in [1]. The correctness of the proposed solution is proved and confirmed by a hardware implementation on FPGA crystals and the corresponding modeling with obtaining time diagrams. These methods are used to check the satisfiability of formulas of three-valued logic. The corresponding logical network is synthesized based on the structure of a formula.

REFERENCES

1. S. L. Kryvyi and V. N. Opanasenko, "Partitioning a set of vectors with nonnegative integer coordinates using logical hardware," Cybernetics and Systems Analysis, Vol. 54, No 2, 310–319 (2018).
2. S. Krivoi, "A criterion of compatibility of systems of linear Diophantine constraints," Lect. Notes Comp. Sci., Vol. 2328, 264–271 (2002).
3. S. L. Kryvyi, "Algorithms for solving systems of linear Diophantine equations in residue fields," Cybernetics and Systems Analysis, Vol. 43, No 2, 3–17 (2007).
4. A. V. Palagin and V. N. Opanasenko, "Reconfigurable computing technology," Cybernetics and Systems Analysis, Vol. 43, No. 5, 675–686 (2007).
5. A. V. Palagin and V. N. Opanasenko, "Design and application of the PLD-based reconfigurable devices," in: M. Adamski, A. Barkalov, and M. Wegrzyn (eds.), Design of Digital Systems and Devices; Lecture Notes in Electrical Engineering, Springer, Berlin-Heidelberg, Vol. 79, 59–91 (2011).
6. V. Opanasenko and S. Kryvyi, "Synthesis of multilevel structures with multiple outputs," in: CEUR Workshop Proceeding of 10th International Conference of Programming (UkrPROG 2016), Vol. 1631, Code 122904, Kyiv, Ukraine (2016), pp. 32–37.
7. Y. P. Kondratenko and E. Gordienko, "Implementation of the neural networks for adaptive control system on FPGA," in: B. Katalinic (ed.), Annals of DAAAM for 2012 & Proc. 23rd DAAAM Intern. Symp. on Intelligent Manufacturing and Automation, Vol. 23, No. 1, Vienna, Austria (2012), pp. 0389–0392.
8. Y. Kondratenko and V. Kondratenko, "Soft computing algorithm for arithmetic multiplication of fuzzy sets based on universal analytic models," in: Information and Communication Technologies in Education, Research, and Industrial Application, Ser. Communications in Computer and Information Science, Vol. 469 (2014), pp. 49–77.
9. A. V. Palagin, V. N. Opanasenko, and S. L. Kryvyi, "Resource and energy optimization oriented development of FPGA-based adaptive logical networks for classification problem," in: V. Kharchenko, Y. Kondratenko, and J. Kacprzyk (eds.), Green IT Engineering: Components, Networks and Systems Implementation, Vol. 105 (2017), pp. 195–218. DOI: DOI 10.1007 978-3-319-55595-9_10.
10. V. N. Opanasenko and S. L. Kryvyi, "Synthesis of neural-like networks on the basis of conversion of cyclic hamming codes." Cybernetics and Systems Analysis, Vol. 53, No. 4, 627-635 (2017).
11. A. Palagin and V. Opanasenko, "The implementation of extended arithmetic on FPGA-based structures," in: Proc. 9th IEEE Intern. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2017) (2017), pp. 1014–1019.
12. J. Drozd, A. Drozd, S. Antoshchuk, A. Kushnerov, and V. Nikul, "Effectiveness of matrix and pipeline FPGA-based arithmetic components of safety-related systems," in: Proc. 8th IEEE Intern. Conf. on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Warsaw, Poland (2015), pp. 785–789.
13. R. Woods, J. McAllister, G. Lightbody, and Yi. Ying, FPGA-Based Implementation of Signal Processing Systems, Wiley, Chichester (2008).
14. V. M. Glushkov, A. A. Letichevskii, and A. B. Godlevskii, Synthesis Methods for Discrete Models of Biological Systems [in Russian], Vyshcha Shkola, Kyiv (1983).
15. J. Anderson, Automata Theory with Modern Applications, Cambridge University Press, Cambridge (2006).
16. J. Lukasiewicz, Aristotle's Syllogistic from the Standpoint of Modern Formal Logic [Russian translation], Inostr. Literatura, Moscow (1959).