mathematics

MDPI

Article

Lattice Structure of Some Closed Classes for Three-Valued Logic and Its Applications

Elmira Yu. Kalimulina †

V. A. Trapeznikov Institute of Control Sciences, Russian Academy of Sciences; elmira.yu.k@gmail.com
† Current address: 65 Profsoyuznaya Street, 117997 Moscow, Russia.

Abstract: This paper provides a brief overview of modern applications of nonbinary logic models, where the design of heterogeneous computing systems with small computing units based on three-valued logic produces a mathematically better and more effective solution compared to binary models. For application, it is necessary to implement circuits composed of chipsets, the operation of which is based on three-valued logic. To be able to implement such schemes, a fundamentally important theoretical problem must be solved: the problem of completeness of classes of functions of three-valued logic. From a practical point of view, the completeness of the class of such functions ensures that circuits with the desired operations can be produced from an arbitrary (finite) set of chipsets. In this paper, the closure operator on the set of functions of three-valued logic that strengthens the usual substitution operator is considered. It is shown that it is possible to recover the sublattice of closed classes in the general case of closure of functions with respect to the classical superposition operator. The problem of the lattice of closed classes for the class of functions T_2 preserving two is considered. The closure operators $\mathcal{R}_1$ for the functions that differ only by dummy variables are considered equivalent. This operator is withiin the scope of interest of this paper. A lattice is constructed for closed subclasses in $T_2 = \{f | f(2, \ldots, 2) = 2\}$, a class of functions preserving two.

Keywords: three-valued logic application; three-valued logic; closure operator; lattice structure; closed subclasses; substitution operator

Citation: Kalimulina, E.Y. Lattice Structure of Some Closed Classes for Three-Valued Logic and Its Applications.

Academic Editor: Michael Voskoglou

Received: 24 November 2021
Accepted: 26 December 2021
Published: 27 December 2021

1. Introduction

The efficiency of a coding scheme per symbol is defined by the relation $y = \log b / b$, where b is the base of a numeral system. It is easy to see that the value of b for which this function is a maximum is $e \approx 2718$. Thus, the optimal base is three because e cannot be used for a rational representation of numbers, and it is quite close to $b = 3$; a ternary system is the most optimal from the point of view of information density [1]. In applications where high data density is of critical importance, the ternary logic gates are the only optimal solution. With ternary logical gates, we can obtain close to an order of magnitude improvement in data density, and a reduction in switching speed by a factor of 13 over the logic of other orders [2].

Extensive research addressing ternary logic has demonstrated that ternary logic gates are the simplest and most economical to implement in comparison to higher-order logic systems, see, for example [3,4]. In these papers, the authors proposed a new approach to design ternary reversible multipliers and demonstrated that ternary computational units are better than other designs, providing about 3% and 11% improvements, respectively, in terms of performance, cost, and energy loss.

Ternary logic systems are optimal for modern computing machines in terms of balance between simplicity and performance. The simplicity of implementation distinguishes ternary logic solutions from those based on other models, for example, from fuzzy logic models, which allow the consideration of a more complete description of systems with more

complicated control. Certainly, strong proven results are available for fuzzy logic [5], which can be much more efficient than classical logic, especially at a high level of implementation, in algorithms for machine learning, artificial intelligence, natural languages processing, image analysis [6], analysis of financial risks [7], and in specialized devices. However, the large number of parameters, states, inputs and outputs, and adding extra variables can exponentially increase the computational complexity of fuzzy logic models. Real-time computing requires special hardware, the development of which is already expensive in contrast to ternary functions. Moreover, one methodology based on carbon nanotube field-effect transistors allows synthesizing inexpensive and low-power consuming circuits exactly for ternary functions [8]. For a four-valued logic function, this process is already somewhat more complicated and becomes more expensive, and research in this direction is also important for the future of computational machines [9,10].

Ternary logic circuits are flexible enough to realize almost all basic logic gates (NAND, NOR, AND, and OR gates) and even some more complicated cascaded logic functions, which are potentially the basis in high-order logic. Ternary logic circuits have advantages in terms of nonvolatility, load capacity, compactness of circuit construction, high efficiency, good robustness with both constant voltages on the input and output, and lacking signal degradation [11].

Ternary logic is the generalization for multivalued logic [12,13]. Furthermore, without loss of generality, instead of multivalued cases, a ternary logic model may be considered. In ternary logic, a statement is assigned one of three values: "true", "false", or "undefined" [1,12,14]; in binary logic, one of two is assigned: "true" or "false". Some features of the operation logic of a ternary computer, for example, the realization of ternary arithmetic operations and, in particular, the representation of negative numbers [15,16], provide possibilities for designing more high-performance modern ternary circuits [17] and devices that will be useful for many modern applications.

Hubs, switches, routers, and other network equipment can be realized on ternary circuits, which will provide notable improvements over binary circuits in terms of interconnects, increases in bandwidth (transmission of trits instead of bits per unit time), and propagation delay [18,19].

Mathematically, ternary logic is more efficient than binary logic [1,12,14,20]. Research and development of algorithms based on three-valued logic are relevant [21], for example, in telecommunications [22,23], in reliability assessment and analysis [24–27], in the symbolic analysis of complex systems [28], in software development and detecting design errors [29], in the field of artificial intelligence (AI) and machine learning [30,31], quantum computing [22,32–34], in e-health systems for the diagnosis of various diseases [35,36], and in physics [37]. This is confirmed by the significant increase in the number of scientific publications in leading scientific journals related to various applications of three-valued logic over the past few years [20].

1.1. A Brief Overview of Nonbinary Logic Model Application

In recent years, three-valued logic has been most applied in the fields of assessing the reliability of complex technical systems and of telecommunications, in the simulation of processes and modern design languages, and in the design of data transmission and processing systems.

In [24], the authors employed a multivalued logic technique to construct explicit functions of the multivalued inputs of the system to analyze its reliability. The various expressions were then compiled in a multivalued Karnaugh map that served as the natural map for a multistate system. Furthermore, the authors showed how to use the multivalued Karnaugh map for coherent multistate system representation to illustrate its features of causality, monotonicity, and relevancy, and to obtain a detailed solution for a standard commodity-supply multistate chain. Additionally, this algebraic technique of multiple-valued logic was applied successfully for reliability analysis of nonrepairable multistate k-out-of-n with multiple levels of system output [25].

The same authors suggested a procedure for converting the minimal symbolic logic expression of the network system's success into a probability-ready expression [28]. This was achieved in terms of multivalued component successes, and it was obtained in minimal form as the disjunction of particular prime implicants or minimal paths of the pertinent network. This function successfully extrapolates the probability-ready expression to the multivalued logical domain. This result is important for applications since it allows the direct transformation of a random logical symbolic expression, on a one-to-one basis,to its statistical expectation form by replacing all logic variables by their statistical expectations, and by substituting arithmetic multiplication and addition for their logical AND and OR operators [28].

In [27], a new mathematical tool for the reliability analysis and evaluation of multistate system reliability was constructed. The proposed method extends a standard Boolean algebra approach used in reliability analysis and allows the construction of the structure function that more exactly describes the investigated system. The authors also developed a new method for the investigation of critical system states and all possible changes to any system component or group of components based on multivalued logical differential calculus.

Additionally, three-valued logic models allow us to take into account qualitative instead of quantitative variables. Quantitative indicators (factors) are discretized by mapping into a certain three-interval scale [38].

This approach allows combining quantitative and qualitative indicators within a single model. The reliability of the factors decreases minimally with such discretization. This allows us to investigate the model as fully as possible. This is especially effective in situations where there it is not possible to quantify the impact of a particular factor on the process. The use of qualitative variables provides additional opportunities for assessing factors.

Simulation is the only available method to check the quality and reliability of complicated and expensive technical systems at their design stage. Automated design tools allow us to assess quality based on real-world operating conditions. Temporary simulation of circuits in an automated simulation system is often based on the principles of three-valued logic.

Ternary logic is effective in constructing computing units for the equipment of data transmission networks. Potentially, the transmission of three states instead of two bits at a time can increase the data transfer rate 1.5 times. With an increase in the number of trits (instead of bits), the speed of the transmission can grow exponentially [23,39,40]. It is possible to implement solutions for data aggregation and transmission based on three-valued logic. These solutions provide a single high-dimensional space for network addressing, both for standard data transmission purposes [41] and for new tasks for controlling robotic devices for the Internet of Things [22]. Several examples of developing a solution for telecommunication systems can be found in [38,42]. An example of building a traffic aggregation scheme based on a k-valued logic model for high-speed transport systems was provided in these papers.

Three-valued logic is also effective for solving both problems of image processing [43] and of cryptography. Quantum computing for data security is the most effective method for protecting mobile robots, the Internet of Things (IoT), and the security of distributed applications, which also uses three-valued logic models. With the rapid advances in quantum computers, ternary computing has become relevant again [33,34,43]. The leading IT companies have introduced their quantum computers operating on several dozens of qubits in the last decade: IBM quantum processors consist of 65 qubits, Google has 72 qubits [44]. The developers plan to release a 1112 qubit processor called Condor by 2023, which should bring quantum technologies to a new commercial level [44].

Additionally, at present, the three-valued logic toolkit is widely used in tasks related to data analysis and the construction of AI models, for example, for tasks of hierarchical data clustering for arbitrary complicated data sets [22,30]. Interpretation models via three-

valued logic allow us to overcome existing limitations on the ability to create fully automatic program-analysis algorithms [45].

2. Theoretical Aspects of Designing Computing Systems Based on Three-Valued Logic

All the applied problems considered above are reduced to the problem of determining the factors that influence the process and considering a countable set P_3 of states of these factors. Any countable number of states can be approximated by three states [46]: 0, 1, and 2 .

For decision making, someone needs to find the value of the output function Y that depends on this set. Accordingly, the output function Y can be represented as a combination of predicates on the set P_3 [47]. For this purpose, complicated predicates and superpositions of these predicates on P_3 are considered.

These predicates can be implemented (from a practical point of view) as circuits of chips, the operation of which is based on three-valued logic.

2.1. Problem of Completeness of Function Classes of Three-Valued Logic

A fundamentally important problem—the problem of completeness of classes of functions of three-valued logic [47]—must be solved to make this implementation possible. From the practical point of view, the completeness of the class of functions guarantees that a circuit with the desired functional diagram can be produced based on an arbitrary finite number of chipsets. For two-valued logic, this problem was solved by Post, which led to the explosive growth of electronics [48].

Post's classical theorem describes five precomplete classes in the set of Boolean functions [48]:

- The class of functions preserving 0;
- The class of functions preserving 1;
- The class of self-dual functions;
- The class of monotone functions;
- Class o linear functions.

For the case of three-valued logic, the problem was solved by Yablonsky in 1958 [46,47]. He proved that there are 18 precomplete classes of functions of three-valued logic:

- classes preserving the sets T_0, T_1, T_2, $T_{0,1}$, $T_{0,2}$, and $T_{1,2}$;
- the class of self-dual functions;
- the class of linear functions;
- classes of monotone functions;
- the class preserving the partition;
- central precomplete classes preserving one of the relations $\left(\begin{smallmatrix}0&1&2&0&0&1&2\\0&1&2&1&2&0&0\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1&2&1&1&0&2\\0&1&2&0&2&1&1\end{smallmatrix}\right)$, $\left(\begin{smallmatrix}0&1&2&2&2&0&1\\0&1&2&0&1&2&2\end{smallmatrix}\right)$;
- Precomplete Slupecki class—the set of all inessential functions.

The precomplete classes for ternary functions differ from precomplete classes for binary functions only by the preservation classes of partitions, central precomplete classes, and the precomplete Slupecki class. Moreover, the preservation classes of partitions are just a method of reducing to the two-valued case. Thus, only the precomplete Slupecki class is fundamentally new for the three-valued logic case [49].

For three-valued logic, it was proved that the completeness problem cannot be solved in a general case [46]: it can be proved only for precomplete classes of the functions [50]. For $k > 2$, there is a continuum set of precomplete classes [51]. If the lattice of closed classes is countable in the case of two-valued logic, then it is exponential (continuous) in the case of three-valued logic. The construction of finite lattice is impossible in this case. However, its closure operators on the set of three-valued logic functions can be considered, which are a strength of the common substitution operator.

Solving the completeness problems for this new closure operator and finding the structure of the lattice of closed classes will help not only to restore the sublattice of closed

classes in the general case of closure of functions with respect to the classical superposition operator, but also to optimize the possible industrial production of chips for functional circuits for solving the problem described in the Introduction.

In the next section, I prove the existence and possibility of constructing a lattice for the closure operator in several lemmas and theorems. The novelty of the obtained results is that a special class of operators on functions of three-valued logic is considered, which is a less general case, but it allows obtaining a finite lattice, which is essential for applications.

3. Main Results

3.1. Lattice of Closed Subclasses T_2 with Respect to $\mathcal{R}_\infty$

Denote P_3 as all 3−valued logic functions. The function $f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n)$ essentially depends on a variable, x_i, if there exists $a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in E_k$, such that $h(x) = f(a_1, \ldots, a_{i-1}, x, a_{i+1}, \ldots, a_n)$ is not constant. This variable x_i is called essential for function f.

Functions f_1 and f_2 are equal if we can derive f_1 from f_2 via deleting essential variables. Let $|X_f|$ be the number of essential variables in f.

Definition 1. *Let M be the set of functions from P_3. Denote $\mathcal{R}_\infty(\mathcal{M})$ as the result of the closure of the set of functions from M with respect to operations of substitution of the function g to the equivalent function $f \sim g$, where*

$$f \sim g \Leftrightarrow \forall \vec{x} \, [\, (f(\vec{x}) = g(\vec{x})) \vee (f(\vec{x}), g(\vec{x}) \in \{0,1\}) \,].$$

Consider a variant of the closure operator $\mathcal{R}_\infty$, for which functions that differ only in dummy variables are considered equivalent. Let us construct a lattice for closed subclasses in $T_1 = \{f | f(1, \ldots, 1) = 1\}$ in the class of functions preserving two.

Definition 2. *Let $f(x_1, \ldots, x_i, \ldots, x_n) \in P_3$, $|X_f| = n$; then, x_i is called $\mathcal{R}_\infty$-essential for f if there are sets $\alpha_1^n = (a_1, \ldots, a_{i-1}, b^1, a_{i+1}, \ldots, a_n)$, $\alpha_2^n = (a_1, \ldots, a_{i-1}, b^2, a_{i+1}, \ldots, a_n)$ such that $f(\alpha_1^n) \sim f(\alpha_2^n)$.*

3.1.1. Completeness in T_2

Definition 3. *The following notation is used:*

$$T^{02} \stackrel{\text{def}}{=} \{f | \exists i \in \{1, X_f\} : \alpha = (a_1, \ldots, a_{X_f}), a_i \in \{0,2\} \Rightarrow f(\alpha) = 2\}$$
$$T^{12} \stackrel{\text{def}}{=} \{f | \exists i \in \{1, X_f\} : \alpha = (a_1, \ldots, a_{X_f}), a_i \in \{1,2\} \Rightarrow f(\alpha) = 2\}$$
$$T^{02} \stackrel{\text{def}}{=} \{f | \alpha = (a_1, \ldots, a_{X_f}); a_i \in \{0,2\}, i \in \{1, X_f\} \Rightarrow f(\alpha) = 2\}$$
$$T^{12} \stackrel{\text{def}}{=} \{f | \alpha = (a_1, \ldots, a_{X_f}); a_i \in \{1,2\}, i \in \{1, X_f\} \Rightarrow f(\alpha) = 2\}$$

Lemma 1. *The class $T^{02}-$ is $\mathcal{R}_\infty$-closed.*

Lemma 2. *The class $T^{12}-$ $\mathcal{R}_\infty$ is closed.*

Proof of Lemma 1. Note that neither the permutation of variables nor the identification or addition of inessential (dummy) variables affects the property functions belonging to the class T^{02}. This follows clearly from the class definitions.

It is also obvious that if $f \in T^{02}$, then for any function $g(f \sim g)$, it is true that $g \in T^{02}$.

Now, I show that the superposition of functions from the class T^{02} will also lie in class T^{02}.

Let $f \in T^{02}, f = f(x_1, \ldots, x_n)$. Consider the function $h = f(g_1, \ldots, g_n)$, where g_i- are either free variables or functions from the set T^{02}.

By contradiction, let $h \notin T^{02}$; then, there is a set $\alpha = (a_1, \ldots, a_{|X_h|}), a_i \in \{0,2\}, 1 \leq i \leq |X_h|$, such that it is true that $h(\alpha) \neq 2$.

By the construction of the function h, and under the condition that $f \in T^{02}$, there is i such that the function $g_i(\beta) \neq$, where $\beta = (b_1, \ldots, b_{|X_{g_i}|}), 1 \leq b_i \leq |X_{g_i}|$ is the projection of vector α on the coordinate axes corresponding to free variables of the function g_i.

Thus, the function $g_i \notin T^{02}$, but this contradicts the choice of function g_i. Thus, $h \in T^{02}$.

Lemma 1 is proved. □

Lemma 2 can be proved by repeating the sketch of the proof of Lemma 1 (by formally replacing T^{02} with T^{12}).

Lemma 3. *The class $T^{02} - \mathcal{R}_\infty$ is precomplete in class T_2.*

Proof. Note that the class $T_2 = \mathcal{R}_\infty(\{,\})$, where $f(|X_f| = 2)\&(f(\alpha) = 2$ if and only if, when $\alpha = (2,2))$, $g(|X_g| = 1)\&(g \in T_2)\&(g \notin T_\sim)$.

Let there be a function $w(w \notin T^{02})$. Then, by definition, there is a set $\alpha = (a_1, \ldots, a_{|X_w|})$, $a_i \in \{0,2\}, 1 \leq i \leq |X_w|$ such that $w(\alpha) \neq 2$.

Let us move from function w to function w', derived from w by identifying variables according to the set α. Namely, the variables in the α set are identified with the same values. Thus, the whole set of variables of the function w may be split into two groups: with respect to 0 and with respect to 2. By identification, that produces the function $w'(|X_{w'}| = 2)\&(w' \notin T^{02})$.

Without loss of generality, let $w'(0,2) = 1$. If this is not true, then, by rearranging the variables and moving to the function $w''(w'' \sim w')$, the function with the specified property can be easily obtained.

If the vector α does not contain elements equal to 2, then the function where $\sim$ is a function w' and satisfies the required properties may be considered.

Note that a function $g(g \in T^{02})\&(|X_g| = 1)\&(g \notin T_\sim)$ exists. Consider a function $w''(w'' \sim w)$ such that:

$$w''(\alpha) = \begin{cases} 1, & \alpha = (2,0) \\ 2, & w'(\alpha) = 2 \\ 0, & \text{otherwise.} \end{cases}$$

Consider a function $v_1(x,y) = g(w''(x,y))$. The property $v_1(\alpha) = 1$ for this function holds if and only if $\alpha = (0,2)$. Additionally, consider a function $v_2 = v_2(x,y) = v_1(y,x)$. It is easy to see that, by construction, it produces $\{v_1, v_2\} \subseteq \mathcal{R}_\infty(\mathcal{T}'^{\in} \cap \sqsupseteq)$.

Consider a function d such that:

$$d(\alpha) = \begin{cases} 2, & a_i \in \{0,2\}, 1 \leq i \leq 2 \\ 1, & \text{otherwise.} \end{cases}, \alpha = (a_1, a_2).$$

It is obvious that $d \in T^{02}$. Let us construct a function m:

$$m(x,y) = d(d(v_1(x,y), d(x,y)), v_2(x,y))$$

$$m(\alpha) = \begin{cases} 2, & a_1 = 1, 1 \leq i \leq 2 \\ 1, & \text{otherwise.} \end{cases}, \alpha = (a_1, a_2).$$

As function $2 \in T^{02}$, a function f can be constructed such that:

$$f(x,y) = m(m(x,2), m(y,2))$$

$$f(\alpha) = \begin{cases} 2, & a_i = 2, 1 \leq i \leq 2 \\ 1, & \text{otherwise.} \end{cases}, \alpha = (a_1, a_2).$$

As mentioned above, $\mathcal{R}_\infty(\{\{,\}\}) = \mathcal{T}_\in$. However, by construction, we can obtain $f \in \mathcal{R}_\infty(\Updownarrow) \subseteq \mathcal{R}_\infty(\sqsupseteq, \mathcal{T}'^\in)$, and, by definition, $g \in T^{02}$; therefore, $T_2 = \mathcal{R}_\infty(\sqsupseteq, \mathcal{T}'^\in)$. The Lemma is proved. □

Lemma 4. *Let $f \in T_2$ and $f \notin T_{01}$. Then, $2 \in \mathcal{R}_\infty(\{)$.*

Proof. Consider the function $h(x) = f(x, \ldots, x)$. It is easy to show that if $h \notin T_{01}$, then $2 \in \mathcal{R}_\infty(\langle)$.

Let $h \in T_{01}$. Note that for any $g(|X_g| = 1)\&(g \in T_{01})$, it holds that $g \in \mathcal{R}_\infty(\langle)$ by condition $f \notin T_{01}$; hence, there is a set $\alpha = (\alpha_1, \ldots, \alpha_n), n = |X_f|, \alpha_i \in \{0,1\}, 1 \le i \le n$ such that $f(\alpha) = 2$. Construct a function $f' = f(g_1, \ldots, g_n)$, $|X_{g_i}| = 1, g_i \in T_{01}, 1 \le i \le n$, at $g_i(0) = \alpha_i$. Note that $\{g_i, h\} \subset \mathcal{R}_\infty(\{)$; therefore, $f' \in \mathcal{R}_\infty(\{)$. Consider a function $h'(x) = f'(x, \ldots, x)$. By construction, it can be obtained that $h'(0) = h'(2) = 2$; therefore, according to the already considered case, we have $2 \in \mathcal{R}_\infty(\langle') \subset \mathcal{R}_\infty(\{)$.

The Lemma is proved. □

Lemma 5. *A class $T_\sim \cap T_2-$ is $\mathcal{R}_\infty-$precomplete in T_2.*

Proof. Let $f \notin T_\sim, f \in T_2, |X_f| = n$. Let me show that $\mathcal{R}_\infty(\{\{ \cup \mathcal{T}_\sim \cap \mathcal{T}_\in\}) = \mathcal{T}_\in$. Note that, by definition, there are at least two sets, $\alpha_1 = (a_1^1, \ldots, a_n^1)$ and $\alpha_2 = (a_1^2, \ldots, a_n^2)$, such that $\alpha_1 \sim \alpha_2$, and $f(\alpha_1) \sim f(\alpha_2)$. Identify variables in f according to the coincidence of identical pairs in vectors α_1 and α_2. Concretely, if $(a_i^1, a_i^2) = (a_j^1, a_j^2)$, then the ith and jth variables are identified. Thus, the function f' of five variables satisfying the following condition was obtained

$$f'(0,1,2,0,1) \sim f'(0,1,2,1,0)$$

Without loss of generality, it can be assumed that after identification variables, the function f' has exactly this order variables. Otherwise, the variables will be reordered. In addition, note that some of the variables of the function f' can be dummy variables.

Note that there is $2, g \in T_\sim \cap T_2(g(0) = 1, g(1) = 0)$. Let us move on from function f' to function $f'' = f'(g(x_1), x_1, 2, x_2, x_3)$, $f'' \in \mathcal{R}_\infty(\mathcal{T}_\sim \cap \mathcal{T}_\in)$. Function f'' satisfies the property

$$f''(0,0,1) \sim f''(1,0,1).$$

Without loss of generality, let $f''(1,0,1) = 2$.

There are functions $f \in T_\sim \cap T_2$, such that $f(\alpha) = 2$ if $\alpha = (2, \ldots, 2)$. Denote the set of such functions as N. By a construction, it is shown that $\mathcal{R}_\infty(\{\{'',\mathrm{N}\}) = T_2$.

Let $h \in T_2$ be an arbitrary function. Consider the functions $g_0, g_1, g_2 \in N(|X_{g_i}| = |X_h| = n)$.

$$g_0(\alpha) = \begin{cases} 2, & \alpha = (2, \ldots, 2) \\ 0, & \text{otherwise.} \end{cases}$$

$$g_1(\alpha) = \begin{cases} 2, & \alpha = (2, \ldots, 2) \\ 1, & \text{otherwise.} \end{cases}$$

$$g_2(\alpha) = \begin{cases} 2, & \alpha = (2, \ldots, 2) \\ 1, & h(\alpha) = 2, \\ 0, & h(\alpha) \neq 2, \end{cases}$$

Consider the function $h'(x_1, \ldots, x_n) = f''(g_2(x_1, \ldots, x_n), g_1(x_1, \ldots, x_n), g_0(x_1, \ldots, x_n))$. By construction, $h' \sim h$. Thereby, $\mathcal{R}_\infty(\langle') = \mathcal{R}_\infty(\langle) \subseteq \mathcal{R}_\infty(\{\{'', \mathcal{N}\}) \subseteq \mathcal{R}_\infty(\{, \mathcal{T}_\sim \cap \mathcal{T}_\in)$. Due to the arbitrariness of the function $h \in T_2$, we obtain $T_2 \in \mathcal{R}_\infty(\{\{, \mathcal{T}_\sim \cap \mathcal{T}_\in\})$.

The Lemma is proved. □

Lemma 6. *A class $T_{01} \cap T_2$ is $\mathcal{R}_\infty$-precomplete in T_2.*

Proof. Consider the function $f \notin T_{01} \cap T_2, f \in T_2$. By Lemma 4, $\mathcal{R}_\infty(\in, \mathcal{T}_{/\infty} \cap \mathcal{T}_\in) \subseteq \mathcal{R}_\infty(\{, \mathcal{T}_{/\infty} \cap \mathcal{T}_\in)$. Let $h \in T_2$ be an arbitrary function from T_2. Note that there is a function $g \in T_{01} \cap T_2$ satisfying the following property:

$$g(0,2) \sim g(1,2)$$

Without loss of generality, $g(1,2) = 2$.

Consider the function $m \in T_{01} \cap T_2(|X_m| = |X_h| = n)$, such that:

$$m(\alpha) = \begin{cases} 2, & \alpha = (2, \ldots, 2) \\ 1, & h(\alpha) = 2, \\ 0, & h(\alpha) \neq 2, \end{cases}$$

The function $h'(x_1, \ldots, x_n) = g(m(x_1, \ldots, x_n), 2)$ satisfies the property $h \sim h'$ by construction. In this way, $h \in \mathcal{R}_\infty(\langle') \subseteq \mathcal{R}_\infty(\{\in, \mathcal{T}_{/\infty} \cap \mathcal{T}_\in\}) \subseteq \mathcal{R}_\infty(\{\{, \mathcal{T}_{/\infty} \cap \mathcal{T}_\in\})$. By the arbitrary function h, we have $T_2 \in \mathcal{R}_\infty(\{, \mathcal{T}_{/\infty} \cap \mathcal{T}_\in)$.

The Lemma is proved. □

Now, it is possible to formulate the main result that follows from these lemmas.

Theorem 1 (Completeness)**.** *There are five precomplete classes in T_2.*

3.2. Completeness Problem for the Operator $\mathcal{R}_\infty$.

Let M be a given set of functions from P_3. Denote the result of the closure of the set of functions M with respect to the operation of substitution and transition of the function g to the equivalent function $f \sim g$ as $\mathcal{R}_\infty(\mathcal{M})$, where

$$f \sim g \Leftrightarrow \forall \vec{x} \, [\, (f(\vec{x}) = g(\vec{x})) \vee (f(\vec{x}), g(\vec{x}) \in \{0,1\}) \,].$$

Consider the following classes: T_{01} is the class of functions preserving the set $\{0,1\}$, T_2 is the function class preserving two, and class $T_\sim$ (also $T_{\{01\},\{2\}}\ (U(R))$ is the function class preserving the relation $\sim$.

It is easy to see that with passing from function f to function g, the property of belonging to classes T_2, T_{01}, $T_\sim$ is preserved. In this way, due to the classes T_2, T_{01}, $T_\sim$ being precomplete with respect to the substitution, and the completion does not add new functions, the following lemma is obtained:

Lemma 7. *Classes T_2, T_{01}, and $T_\sim$ are $\mathcal{R}_\infty$-precomplete.*

Lemma 8. *Let $f \notin T_{01}$, then $2 \in \mathcal{R}_\infty(\{)$.*

Proof. It is easy to check that if $h(x) \notin T_{01}$ is a 1-place function, then $2 \in \mathcal{R}_\infty(\langle)$. Thus, if the function $g(x) = f(x, \ldots, x), g \notin T_{01}$, then the Lemma is proved.

Let $g \in T_{01}$. By condition, there is a set $\vec{\alpha} = (a_1, \ldots, a_n), a_i \in \{0,1\}$ such that $f(\vec{\alpha}) = 2$. Consider a function f' such that

$$g'(x) = f(g_1(x), \ldots, g_n(x))$$

where $g_i(g_i(0) = a_i)\&(g_i \sim g)$. Notice that $g' \in \mathcal{R}_\infty(\{)$ and $g'(0) = 2$, then $2 \in \mathcal{R}_\infty(\}') \subseteq \mathcal{R}_\infty(\{)$.

The Lemma is proved. □

Theorem 2 (Completeness)**.** *There are three $\mathcal{R}_\infty-$precomplete classes T_2.*

4. Conclusions

The proved results can be interpreted in the following way for applications: Suppose we have a finite number of microcircuit industrial manufacturers. These three-valued logic microcircuits can be considered as the class of three-valued functions (basic operations). The presented results indicate that any other compound function (logic formula) that can be expressed via functions from this class also belongs to this class. This compound function is also called a superposition of basic functions (operations). The study of the superposition of functions defined on a finite set led to the emergence of the theory of closed classes. The term closed class means precisely closed in superposition. The initial problem for the theory of closed classes is the problem of functional completeness [49]. It consists of determining, for an arbitrary set of functions, whether it is possible to obtain all functions with arguments and values in this set from these functions by superposition. I proved that it is possible. Moreover, we can describe all of them explicitly (via lattice construction).

If there is a finite set of three-valued logic microcircuits, the solution for the problem of completeness of closed classes means that we can construct all possible functional schemes for a specific application problem by combining different microcircuits from this finite set. Earlier, similar results were considered for a binary logic case [52], where the theoretical foundation for designing current circuits in Boolean and linear algebra was provided. However, for the case of multivalued logic (in particular, for three-valued logic), this is still an open problem. Different techniques and tools have been applied to this problem, but the current results mainly include either the description of general approaches or attempts to apply the mathematical tool of linear algebra (nonlogical tools) to synthesize nonbinary digital current circuits [53], or the more complicated pure theoretical tools that are impossible to realize [54], or the realization for some special operators is only given [55], or a complete superposition calculus is only provided for first-order-type logic [56]. I proved the result for the three-valued case and showed that any logic function (digital circuit) can be synthesized from a finite number of different microcircuits. My results are based on the closure operators on the set of functions of three-valued logic, which is a strength of the usual substitution operator. I proved that the completeness problem for this operator has a solution. Therefore, it is possible to recover the sublattice of closed classes in the general case of the closure of functions with respect to the classical superposition operator.

Three-valued logic can potentially produce considerable improvements in the field of computer networks and data networks. Research is in progress in the field of nonclassical approaches to logic synthesis and circuit design of digital IP modules for computer technology, and control and communication systems [38,41,42,57].

Let me separately highlight two examples where the results obtained in this paper were directly applied, and are of greatest interest to me:

Example 1. *The problem of traffic aggregation in cellular networks for providing Internet access on high-speed trains. In [38], it was shown how to implement a new traffic aggregation protocol based on predicates of three-valued logic exactly on these results. A new working scheme of LTE modems was also described: coding three states instead of two (and, accordingly, using all their possible combinations) allows covering all possible states of a data transmission device and, as a result, processing more information per unit of time.*

In general, a large number of states can be considered for an LTE modem, for example:

- Modem is off,
- Modem is on,
- Low latency for a pocket transmission,
- A good coverage of a cellular operator network, etc.

In reality, a countable number exists; however, it is possible to approximate all states to three fundamentally important states from the point of view of applications:

1, The modem is working, and traffic is transmitted with normal latency;

2, The modem does not work, or
0, The state is not recognized.

The functional elements of an LTE modem model may be represented by logical units that generate multivalued signals at the inputs/outputs. These signals are discretized by three values, for example, $\{A, B, \hat{x}\}$, where A, B are constants, and $\hat{x}$ is a positive signal. Then, the operation of logical units with such signals may be described, for example, by the following tables (that are analogues of truth tables in binary logic), where, for simplicity, the constants are denoted by 0 and 1, and the positive signal as 2.

A	$A+1$
ine 0	1
ine 1	2
ine 2	0

A	$A+2$
ine 0	2
ine 1	0
ine 2	1

$m(A,B)$	0	1	2
ine 0	0	0	0
ine 1	0	1	1
ine 2	0	1	2

$m_1(A,B)$	0	1	2
ine 0	0	1	2
ine 1	1	1	1
ine 2	2	1	1

$m_{01}(A,B)$	0	1	2
ine 0	0	1	1
ine 1	1	0	1
ine 2	2	0	0

Furthermore, we can construct and calculate all possible values for the resulting function, for example, for the following:

$$f = m_1(m(X_1; m_{01}(X_1; X_2)); A + 2(X_1)).$$

The table below shows an example of calculating values for this resulting function.

$f(A,B)$	0	1	2
ine 0	2	2	2
ine 1	1	2	1
ine 2	1	2	2

Figure 1 demonstrates a logical diagram of the construction of this function.

A similar approach based on the completeness problem of three-valued logic but for a different task was used in [2]. The authors also implemented considered schemes and performed a simulation that demonstrated a reduction in switching speed by a factor of 13 compared to classical results.

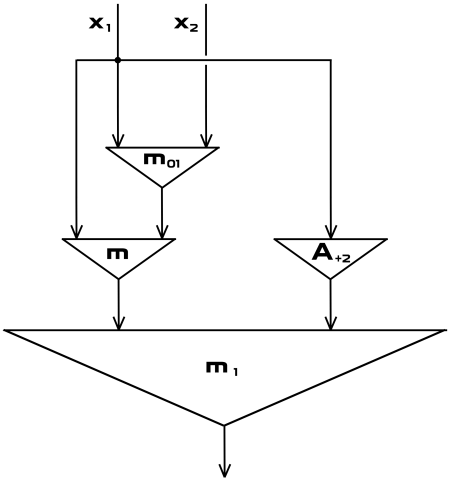

Figure 1. An example of the resulting function construction $f(A, B)$.

Example 2. *The construction of structural reliability models of complex systems for which there is no strict concept of failure. During modeling, three primitive logic functions describing three states of the system's unit are considered: an operable state, a valid state, and a failure. Then, these are used to build a combination of any other state as the ternary predicate as well as handling the ternary the whole system state. Th reliability functions are also written as a three-valued logic predicate [58].*

Many fuzzy factors that exist in the traffic aggregating scheme can be discretized. To establish a functional relationship between the input and output parameters of the system (i.e., the performance and reliability of the entire system), we can construct a structural function that establishes a one-to-one correspondence between the possible states of n elements and the state of the entire system:

$$\varphi(x_1, \ldots, x_n) = \varphi(x) : \{0, 1, 2\}^n \to \{0, 1, 2\}.$$

Figure 2 demonstrates a general graphical interpretation of the structural function of the process for assessing the quality (performance and reliability) of the above-mentioned traffic aggregation system (the detailed scheme is given in [38]). When constructing this scheme, the following key internal and external factors that affect the traffic transmission were analyzed and selected: the quality of data transmission (availability of access to the external network through one of the three cellular communication channels), equipment quality, and power supply stability.

To demonstrate how to construct such a scheme, only the main influencing factors were selected. With the assistance of software for the automated design of such models, and by taking into account additional parameters, the list can be significantly expanded, for example, by adding the speed of the object, the signal level of cellular stations, the amount of network load (number of users), channel capacity, types of traffic, etc. Let us consider the following factors for the traffic aggregation problem:

Factor	Factor and its description		
	0	1	2
ine Data link 1 (x_1)	Not available	Weak signal	Stable connection
ine Data link 2 (x_2)	Not available	Weak signal	Stable connection
ine Data link 3 (x_3)	Not available	Weak signal	Stable connection
ine x_{11}	Not available	Weak signal	Stable connection
ine Power supply (x_4)	failure	Unstable operation	Working state
ine LTE modem (x_5)	failure	Unrecognized	Working state
ine x_{12}	failure	Unstable operation	Working state
ine The state of the entire system (Y)	failure	Unstable network access	Working state

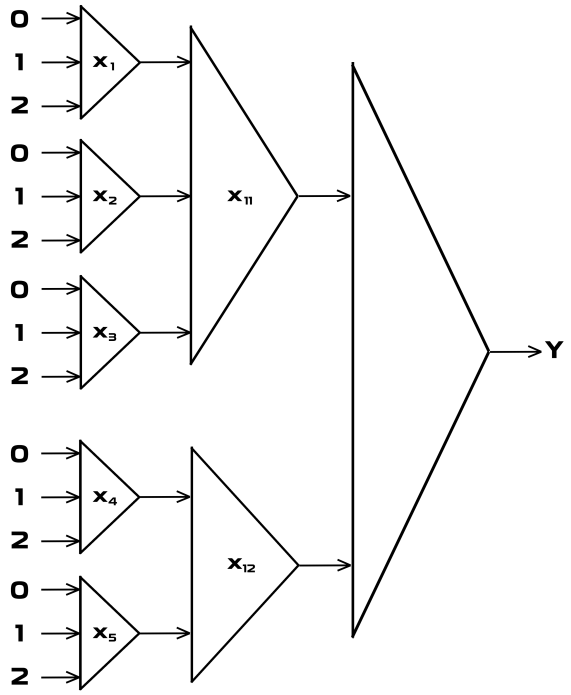

Figure 2. Diagram of the structural function of the quality assessment of a traffic aggregation system.

In order to reduce the computational complexity of the structural function, all variables may be grouped into classes of similarity, for example, data transmission links(x_{11}) and network equipment (x_{12}); and the intermediate result calculating for these classes may be based on expert assessments. The intermediate results' calculation is described in the following tables.

x_1	0	0	0	0	0	0	0	0	0
ine x_2	0	0	0	1	1	1	2	2	2
ine x_3	0	1	2	0	1	2	0	1	2
ine x_{11}	0	0	2	0	1	2	2	2	2

x_1	1	1	1	1	1	1	1	1	1
ine x_2	0	0	0	1	1	1	2	2	2
ine x_3	0	1	2	0	1	2	0	1	2
ine x_{11}	0	1	2	1	1	2	2	2	2

x_1	2	2	2	2	2	2	2	2	2
ine x_2	0	0	0	1	1	1	2	2	2
ine x_3	0	1	2	0	1	2	0	1	2
ine x_{11}	2	2	2	2	2	2	2	2	2

x_4	0	0	0	1	1	1	2	2	2
ine x_5	0	1	2	0	1	2	0	1	2
ine x_{12}	0	0	0	0	1	2	0	2	2

Then, the output function Y may be derived from these intermediate values:

x_{11}	0	0	0	1	1	1	2	2	2
ine x_{12}	0	1	2	0	1	2	0	1	2
ine Y	0	0	0	0	1	2	0	2	2

Thus, the result of the structural function construction of the system is a vector Y, interpreted as a column of the truth table on ordered sets of variables:

$$\{00, 01, 02, 10, 11, 12, 20, 21, 22\} \rightarrow Y.$$

The obtained structural function can be used to determine the efficiency and reliability of the object under study for the given input parameters. Furthermore, this can be detailed by adding new predicates for aggregation, for example (denoted with special names for practical clarity) [38],

$simCardOp1Slot(x_1, ..., x_n)$: the value of the function determines if the SIM card is in the corresponding slot or not, in the aggregator;

$latency(x_1, \ldots, x_n)$: indicates if the latency of the channel satisfies the requirements;

$modemState(x_1, \ldots, x_n)$: if the modem works or not;

$coverageOperator_1(x_1, \ldots, x_n)$: indicates if modems are located in the coverage area of Operator_1;

$coverageOperator_k(x_1, \ldots, x_n)$: the modems in the coverage area of Operator_k, etc.

The proved results make it possible to construct the superposition of these functions as a complex predicate for traffic aggregation:

$$trafficAggr\Big(latency(x_1, \ldots, x_n), modemState(x_1, \ldots, x_n), ..., coverageOperator_k(x_1, \ldots, x_n)\Big).$$

The constructed structure function can be used to determine the efficiency and reliability of the system for given input parameters. Further research and development of this technique will allow a more complete assessment of the effects of various factors on the performance of the system as a whole.

A similar process of the logical synthesis and circuit design of double- and multivalued digital nodes was described in [57], demonstrating the advantages of the considered architectural and circuitry solutions for the synthesis of double- and multivalued digital nodes of different applications in comparison to double-valued Boolean algebra. However, these results describe the concept rather than providing a method of synthesizing circuits from a practical point of view. The results obtained in this paper can be used to optimize the industrial production of chipsets (or microcircuits) for new functional circuits for transmission and data processing tasks. The problem of constructing a lattice of closed classes is, in some sense, an inverse problem to the problem of completeness, and it provides an explicit answer on how to implement some functional schema from a finite number of microcircuits.

In general, many results obtained for the 3-valued case (and for k-valued), are a generalization of the results for the two-valued case. But there is the main fundamental difference between P_2 and P_3 (more general for P_k) is that the class P_k ($k > 2$) is larger than P_2. And with k increase, the class P_k is so large that its full description is impossible. For a given predicate we can check that the closed class is finite. And for some given finite sets of functions, we can describe the closed class generated by them via predicates.

These problems can be solved easily only for P_2 and for P_3. And only for these cases, we can construct the lattice of all closed classes [49]. But for higher-order cases ($k > 3$) these problems are not solvable: there is no good description of all closed classes (for P_k, $k > 3$) that allow solving practical computational tasks.

Funding: This publication was prepared with the support of the Russian Foundation for Basic Research according to the research project No. 20-01-00575 A.

Institutional Review Board Statement:

Informed Consent Statement:

Data Availability Statement:

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

References

1. Jeff, C. Ternary Computing Testbed 3-Trit Computer Architecture. Ph.D. Thesis, Computer Engineering Department, California Polytechnic State University, San Luis Obispo, CA, USA, 2008; p. 192. Available online: Http:.Xyzzy.Freeshell.Org/Trinary/Cpe%20report%20-%20ternary%20computing%20testbed%20-%20rc6a.Pdf (accessed on).
2. Wang, X.Y.; Zhou, P.F.; Eshraghian, J.K.; Lin, C.Y.; Iu, H.H.C.; Chang, T.C.; Kang, S.M. High-density memristor-cmos ternary logic family. *IEEE Trans. Circuits Syst. Regul. Pap.* **2020**, *68*, 264–274.
3. Asadi, M.A.; Mosleh, M.; Haghparast, M. Toward novel designs of reversible ternary 6:2 Compressor using efficient reversible ternary full-adders. *J. Supercomput.* **2021**, *77*, 5176–5197. https://doi.org/10.1007/s11227-020-03485-7
4. Asadi, M.A.; Mosleh, M.; Haghparast, M. Towards designing quantum reversible ternary multipliers. *Quantum Inf. Process.* **2021**, *20*, 226. https://doi.org/10.1007/s11128-021-03161-6
5. Novak, V.; Perfilieva, I.; Mockor, J. *Mathematical Principles of Fuzzy Logic*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 517.
6. Mikhov, D.; Kondratenko, Y.; Kondratenko, G.; Sidenko, I. Fuzzy Logic Approach to Improving the Digital Images Contrast. In Proceedings of the 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, 2–6 July 2019; pp. 1183–1188. doi:10.1109/UKRCON.2019.8879961
7. Mirshahi, S.; Novak, V. A Fuzzy Approach for Similarity Measurement in Time Series, Case Study for Stocks. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*; Springer: Cham, Switzerland, 2020; Volume 1239. https://doi.org/10.1007/978-3-030-50153-2_42.
8. Kim, S.; Lee, S.-Y.; Park, S.; Kim, K.R.; Kang, S. A Logic Synthesis Methodology for Low-Power Ternary Logic Circuits. *IEEE Trans. Circuits Syst. Regul. Pap.* **2020**, *67*, 3138–3151. doi:10.1109/TCSI.2020.2990748.
9. Choi, B. Designing the First Many-valued Logic Computer. *Int. J. Mech. Eng. Robot. Res.* **2017**, *6*.
10. Ciuni, R.; Ferguson, T.M.; Szmuc, D. Modeling the Interaction of Computer Errors by Four-Valued Contaminating Logics. In *Logic, Language, Information, and Computation 2019*; Iemhoff, R., Moortgat, M., de Queiroz, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11541. https://doi.org/10.1007/978-3-662-59533-6_8.
11. Zhang, H.; Zhang, Z.; Gao, M.; Luo, L.; Duan, S.; Dong, Z.; Lin, H. Implementation of unbalanced ternary logic gates with the combination of spintronic memristor and CMOS. *Electronics* **2020**, *9*, 542.
12. Trogemann, G.; Nitussov, A.Y.; Ernst, W. *Computing in Russia: The History of Computer Devices and Information Technology Revealed*; Vieweg+Teubner Verlag: 2001; pp. 19, 55, 57, 91, 104–107, ISBN 978-3-528-05757-2.
13. Rumyantsev, D. Interviews with the Designer of the Ternary Computer. Upgrade 33:175 (August 2004). An Interview with Nikolai Brusentsov, Designer of the Setun Ternary Computer. (In Russian)
14. The Ternary Calculating Machine of Thomas Fowler. Available online: www.Mortati.Com/Glusker/Fowler/Index.Htm (accessed on).
15. Parhami, B.; McKeown, M. Arithmetic with binary-encoded balanced ternary numbers. In Proceedings of the 2013 Asilomar Conference on Signals, Systems and Computers, 2013; pp. 1130–1133, doi:10.1109/ACSSC.2013.6810470.
16. Asibelagh, A.G.; Mirzaee, R.F. Partial Ternary Full Adder versus Complete Ternary Full Adder. In Proceedings of the 2020 International Conference on Electrical Communication and Computer Engineering (ICECCE), 2020; pp. 1–6.
17. Luo, L.; Dong, Z.; Hu, X.; Wang, L.; Duan, S. MTL: Memristor Ternary Logic Design. *Int. J. Bifurc. Chaos* **2020**, *30*, 2050222.
18. Jaber, R.; Elhajj, A.; Nimri, L.; Haidar, A. A Novel Implementation of Ternary Decoder Using CMOS DPL Binary Gates. In Proceedings of the 2018 International Arab Conference on Information Technology (ACIT), 2018; pp. 1–3. doi:10.1109/ACIT.2018.8672698.
19. Jaber, R.; Haidar, A. Multiple-Valued Logic Circuit Design and Data Transmission Intended for Embedded Systems. 2020. doi:10.13140/RG.2.2.27826.32961.
20. Subhash, K. On Ternary Coding and Three-valued Logic. *arXiv* **2018**. arXiv:1807.06419.
21. Cobreros, P.; Égré, P.; Ripley, D.; Van Rooij, R. Three-valued Logics And Their Applications. *J. Appl.-Non-Class. Logics* **2014**, *24*, 1–11. doi:10.1080/11663081.2014.909631.

22. Bykovsky, A.Y. Heterogeneous Network Architecture for Integration of Ai and Quantum Optics by Means Of Multiple-valued Logic. *Quantum Rep.* **2020**, *2*, 126–165. doi:10.3390/Quantum2010010.
23. Jin, Y.; He, H.; Lü, Y. Ternary Optical Computer Architecture. *Phys. Scr.* **2005**, T118. doi:10.1238/Physica.Topical.118a00098.
24. Rushdi, A.M.A.; Al-Amoudi, M.A. Reliability analysis of a multi-state system using multi-valued logic. *IOSR J. Electron. Commun. Eng. (IOSR-JECE)* **2019**, *14*, 1–10.
25. Rushdi, A.M.A. Utilization of symmetric switching functions in the symbolic reliability analysis of multi-state k-out-of-n systems. *Int. J. Math. Eng. Manag. Sci. (IJMEMS)* **2019**, *4*, 306–326.
26. Ren, Y.; Zeng, C.; Fan, D.; Liu, L.; Feng, Q. Multi-State Reliability Assessment Method Based on the MDD-GO Model. *IEEE Access* **2018**, *6*, 5151–5161, doi:10.1109/ACCESS.2018.2789931.
27. Zaitseva, E.; Vitaly, L. Reliability analysis of multi-state system with application of multiple-valued logic. *Int. J. Qual. Reliab. Manag.* **2017**, *34*, 862–878.
28. Rushdi, A.M.A.; Amashah, M.H. Symbolic Reliability Analysis of a Multi-State Network. In Proceedings of the 2021 National Computing Colleges Conference (NCCC), 2021; pp. 1–4, doi:10.1109/NCCC49330.2021.9428876.
29. Liu, S.; Shi, Y.-F.; Huang, M.-Y. Model checking software product line based on multi-valued logic. *Int. J. Reliab. Saf.* **2018**, *12*, 364–393.
30. Aizenberg, I. *Complex-Valued Neural Networks with Multi-valued Neurons*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2011; Volume 353. doi:10.1007/978-3-642-20353-4.
31. Al-Askaar, S.; Marek, P. A New Approach to Machine Learning Based on Functional Decomposition of Multi-Valued Functions. In Proceedings of the 2021 IEEE 51st International Symposium on Multiple-Valued Logic (ISMVL), 2021.
32. Hu, Z.; Deibuk, V. Design Of Ternary Reversible/Quantum Sequential Elements. *J. Thermoelectr.* **2018**, *1*, 5–16.
33. Muthukrishnan, A.; Stroud, C.R., Jr. Multivalued Logic Gates For Quantum Computation. *Phys. Rev. A.* **2000**, *62*. doi:10.1103/Physreva.62.052309.
34. Muthukrishnan, A. Classical and Quantum Logic Gates: An Introduction to Quantum Computing—Rochester Center For Quantum Information (Rcqi). *Quantum Inf. Semin.* **1999**, 22.
35. Kacem, S.B.H.; Borgi, A.; Othman, S. DAS-Autism: A Rule-Based System to Diagnose Autism Within Multi-valued Logic. In *Smart Systems for E-Health*; Springer: Cham, Switzerland, 2021; pp. 183–200.
36. Duer, S.; Bernatowicz, D.; Wrzesień, P.; Duer, R. Examination of informativeness of diagnoses expressed with multiple-valued logic. *Biul. Wojsk. Akad. Tech.* **2018**, *67*. doi:10.5604/01.3001.0012.0992.
37. Warzecha, M.; Oszajca, M.; Pilarczyk, K.; Szaciłowski, K. A Three-valued Photoelectrochemical Logic Device Realising Accept Anything And Consensus Operations. *Chem. Commun.* **2015**, *51*, 3559–3561. doi:10.1039/C4cc09980j.
38. Kalimulina, E.Y. Application of multi-valued logic models in traffic aggregation problems in mobile networks. In Proceedings of the 2021 IEEE 15th International Conference on Application of Information and Communication Technologies (AICT), 2021; pp. 1–6, doi: 10.1109/AICT52784.2021.9620244.
39. Gaudet, V. A Survey and Tutorial on Contemporary Aspects of Multiple-valued Logic and Its Application to Microelectronic Circuits. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2016**, *6*, 5–12. doi:10.1109/Jetcas.2016.2528041.
40. Wu, H.; Bai, Y.; Li, X.; Wang, Y. Design of High-speed Quaternary D Flip-flop Based on Multiple-valued Current-Mode. *J. Phys. Conf. Ser.*. **2020**, *1626*. doi:10.1088/1742-6596/1626/1/012067.
41. Esin, A.; Yavorskiy, R.; Zemtsov, N. Brief Announcement Monitoring Of Linear Distributed Computations. In *Distributed Computing*; Dolev, S., Ed.; Lecture Notes in Computer Science; Vol 4167. Springer: Berlin/Heidelberg, Germany, 2006; Volume 4167. doi:10.1007/11864219-47.
42. Kalimulina, E.Y. Construction of a lattice structure of some closed classes in four-valued logic with application to traffic transmission problem. In Proceedings of the International Conference on Computing and Communication Networks (ICCCN), 2021.
43. Ebrahim, A.; Abdolreza, D.; Sanaz, S. Design Of Multiple-valued Logic Gates Using Gate-diffusion Input For Image Processing Applications. *Comput. Electr. Eng.* **2018**, *69*, 142–157. doi:10.1016/J.Compeleceng.2018.05.019.
44. IBM Quantum Summit 2020: Exploring the Promise of Quantum Computing for Industry. Available online: www.Ibm.Com/Blogs/Research/2020/09/Quantum-industry/ (accessed on).
45. Reps, T.W.; Sagiv, M.; Wilhelm, R. Static Program Analysis Via 3-valued Logic. In *Computer Aided Verification*; Alur, R., Peled, D.A., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3114. doi:10.1007/978-3-540-27813-9_2.
46. Yablonskii, S.V. Functional Constructions in A K-valued Logic, Collection of Articles on Mathematical Logic and Its Applications to Some Questions of Cybernetics. *Trudy Mat. Inst. Steklov. Acad. Sci. Ussr, Mosc.* **1958**, *51*, 5–142.
47. Yablonskiy, S.V.; Gavrilov, G.P.; Kudryavtsev, V.B. Logical Algebra Functions And Post Classes. Nauka: Moscow, Russia, 1966.
48. Post, E.L. *Two-Valued Iterative Systems of Mathematical Logic*; Annals of Mathematics Studies Princeton University Press: 1941; Volume 5.
49. Zhuk, D.N. From two-valued logic to k-valued logic, Intelligent systems. *Theory Appl.* **2018**, *22*, 131–149.
50. Yablonsky, S.V. On functional completeness in three-valued calculus // Dokl. Academy of Sciences of the USSR. -1954. -T. 95, No. 6. -S. 1152-1156
51. Yanov, Y.I.; Muchnik, A.A. On the existence of k-valued closed classes without a finite basis. *Dokl. Acad. Sci. Ussr* **1959**, *127*, 44–46.

52. Butyrlagin, N.V.; Chernov, N.V.; Prokopenko, N.V.; Bugakova, A.V. Current Digital Logical Elements' Synthesis and Circuitry: Linear Threshold Approach. In Proceedings of the International Seminar on Electron Devices Design and Production (SED), Prague, Czech Republic, 23–24 April 2019.
53. Naware, N.K.; Khurge, D.S.; Bhandari, S.U. Review of Quaternary Algebra and Its Logic Circuits. In Proceedings of the 2015 International Conference on Computing Communication Control and Automation, 2015; pp. 969–973, doi:10.1109/ICCUBEA.2015.204
54. Kishor, B.; Singh, A.K.; Bandewar, S. Implementation of Trinary/Quaternary Addition using Multivalue Logic Digital Circuit. *Int. J. Comput. Appl.* **2015**, *118*, 4.
55. Prokopenko, N.N.; Chernov, N.I.; Yugai, V.; Butyrlagin, N.V. The element base of the multivalued threshold logic for the automation and control digital devices. In Proceedings of the 2017 International Siberian Conference on Control and Communications (SIBCON), 2017; pp. 1–5, doi:10.1109/SIBCON.2017.7998508
56. Nummelin, V.; Bentkamp, A.; Tourret, S. Vukmirovic P, Superposition with First-class Booleans and Inprocessing Clausification. In Automated Deduction—CADE 28; Platzer, A., Sutcliffe, G., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2021; Volume 12699. https://doi.org/10.1007/978-3-030-79876-5_22.
57. Prokopenko, N.N.; Butyrlagin, N.V.; Chernov, N.I.; Yugai, V.Y. The linear concept of logical synthesis of digital IP-modules of control and communication systems. In Proceedings of the 2015 International Siberian Conference on Control and Communications (SIBCON), 2015; pp. 1–7, doi: 10.1109/SIBCON.2015.7147182.
58. Karasev, V. Theoretical Foundations of Ternary Logic Application in Logical-Probabilistic Method for Assessment of Socio-Economic System Reliability. In Proceedings of the 2021 62nd International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS), 2021; pp. 1–4, doi:10.1109/ITMS52826.2021.9615326.