ISSN 1814-4225 (print)
ISSN 2663-2012 (online)

UDC 004.8:004.94:004.67 doi: 10.32620/reks.2025.1.07

Zoya DUDAR, Volodymyr LIASHYK

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

METHOD FOR SOLVING QUANTIFIER LINEAR EQUATIONS BASED ON THE ALGEBRA OF LINEAR PREDICATE OPERATIONS

*The **subject** involves structured approaches that extend the existing set of mathematical tools for processing complex relationships within databases and computational systems. This is particularly relevant for applications requiring efficient information retrieval, knowledge representation, and logical inference in automated decision-making environments. The **task** of this article is to develop a method for solving quantifier linear equations using the algebra of linear predicate operations, aimed at improving database query optimization and enhancing the capabilities of intelligent systems. The **methods** used in this research include algebraic techniques, logical operations, and matrix-based transformations to model and efficiently solve the predicate equations. By leveraging the algebra of finite predicates, the proposed approach enables a more systematic and scalable way to handle logical dependencies and optimize computational workflows. The method integrates linear logical operators, ensuring that complex queries and constraints in databases can be represented and processed through formal mathematical models. Additionally, it introduces a framework that enhances the structural representation of knowledge, facilitating intelligent data analysis. Because of the study, a formal method was developed to solve quantifier linear equations, enabling more effective query optimization, logical reasoning, and decision-support mechanisms within expert and automated information systems. The research demonstrates that algebraic approaches can significantly improve the efficiency of information retrieval processes, particularly in intelligent databases where relational constraints and dependencies play a crucial role. Benchmarks conducted on synthetic datasets validate the scalability of the method, showing that it maintains linear execution time growth even with increasing data complexity. **Conclusion:** the proposed method expands the mathematical foundation for solving logical equations in computational environments, providing a powerful tool for intelligent systems and database optimization. The ability to formalize and process complex logical relationships contributes to improved decision-making accuracy and automation efficiency.*

Keywords: *quantifier linear equations; algebraic methods; formal modeling; logical methods; algebra of finite predicates; predicate equation.*

Introduction

Motivation

Advances in computational technology, particularly in microelectronics and computer architecture, have led to significant technical capabilities in modern computing machines, such as high speed and large memory capacity. This has facilitated the expansion of the range of problems that can be solved using computers and increased their role in human life. However, this progress is purely quantitative. Simple enhancement of computer functions is effective only when humans can service them; otherwise, such enhancement becomes pointless.

The rapid development of computing technology and its widespread application drives high rates of development in methods for creating intelligent systems (IS) for various purposes [1]. Today, methodological and technical approaches for creating and using information systems have already been developed [2, 3, 4]. Modern intelligent information systems can perform functions previously considered exclusively human prerogatives, such as proving mathematical theorems, translating texts from one language to another, diagnosing diseases, and performing many other functions. All these problems cannot be solved without the involvement of a universal mathematical language.

This article aims to present the theoretical basis and practical applications of the method, illustrating how predicate equations can be used to simplify data modelling and enhance the efficiency of data analysis processes. By bridging the gap between theory and practice, this work demonstrates the applicability of predicate equations to real-world problems, offering new opportunities for the systematic exploration and understanding of complex datasets.

State of the art

The ability of a computer to provide a high degree of information processing is becoming increasingly important. The insufficiency of computing machine software has shown the necessity of creating intelligent systems that can assist in designing and creating software products. The form in which this information is represented on the computer significantly impacts the speed and quality of information processing by intelligent systems [1]. Various information systems use different methods for representing knowledge depending on the specific application areas. Knowledge representation involves formalizing beliefs through records or languages. A formalization that is perceived by the computer is particularly interesting, for which formal languages are developed to represent knowledge in the computer's memory.

Recently, numerous practical applications of modern abstract algebra have been found in databases and intelligent systems, which has led to increased interest in the possibilities of the algebraic description of information [2]. Various high-level language translators and algorithmic algebras were developed on the basis of algebraic methods in programming theory.

The automation of software development and the design of computer systems is an important and urgent problem in computer technology [3]. One of the main tasks of this theory is the problem of the optimal translator from one language to another, which consists of finding the optimal implementation of the algorithm in another language. The use of algebraic methods allows you to create effective algorithms for translating and optimizing programs.

The algebraic approach to the description of derived information distinguishes a certain algebraic system – the algebra of queries, in terms of which derived information is written through the basic one [4]. This allows you to formalize the processes of processing requests and obtaining information in databases. When designing relational databases, knowledge about the subject area is presented in the form of relations, which is effective when designing expert systems.

Thus, it is necessary to translate from the programming language to the machine language with simultaneous optimization of the source program. The process of solving such a problem is divided into several intermediate stages, each of which involves a partial optimization of the algorithm and translation into an intermediate language corresponding to this stage. A database is an information system that stores and processes information and can provide answers to requests. Moreover, it should be possible to obtain not only information directly stored in the database but also derivative information obtained based on basic information. The task of obtaining derivative information is directly related to the task of the result in intelligent systems [5].

Database queries can be written using the formulas of some logical languages, for example, using the language of the difference of statements or the difference of predicates of different orders, and the expressive possibilities of these differences are different [6]. There are various differences between classical and non-classical logic. Boolean algebras, for example, correspond to the classical difference in statements, and special Geiting algebras correspond to the intuitionistic differences' statements.

To be able to mathematically describe the functions of the intellect, it was necessary to create a formal language that could be used to conduct such a description. The formal language had to be chosen in such a way that any finite alphabetic operator could be written down in a convenient form. The algebra of finite predicates described below is such a language.

Predicate algebra describes only knowledge about facts Algebra of operations on predicates or the algebra of predicate operations must formalize operations on knowledge [7] presented as a relation on some object space. The algebra of predicates determines the declarative component of knowledge, and algebraic predicate operations a procedural component of knowledge [8]. Two types of facts are distinguished: the first describes the connection of two entities; moreover, one of them will be defined as a subject, and the second - as an object predicate action. In the first case, the fact is a triplet "subject – predicate – object", in which the predicate is relational, and the subject and object indicate two subjects. The second type of fact is a triplet "subject – attribute – value", where subject – this is an object about which a fact is recorded, an attribute-named feature of an object that has a certain in advance property, and the value is some value of this feature, the scope of which can be in some cases known For example, these can be attributed facts place and time of a certain action. The facts of the second type allow you to split a set of entities on equivalence classes and narrow the search space ways of conclusion. To obtain such triples, there is a definition of the entities that form many approaches. The task is set as a task of predicting the relationship between a pair of entities, which determines whether a couple is connected entities through relation.

In connection with the constant increase in the degree of computerization, developments in this direction are considered, which have an urgent need for a new theoretical and practical base in the field of the formal description of excellent physical information objects. Expectations that the role of a universal information mediator would improve language programming were not fulfilled, it became clear that in terms of convenience and flexibility, no artificial language can compare with natural language [9]. At this time, methodological and technical approaches to the creation and use of information systems have already been developed. Currently, available intelligent information systems can violate functions that were previously considered an exclusive prerogative of humans: prove mathematical theorems, translate texts from one language to another, diagnose diseases, and affect many other functions.

Another direction of informatization is the creation of a system of integrated knowledge [10] and the development of methods of active, mental navigation in these systems, including through global computer networks. Currently, the problem of hardware and software tools that effectively manipulate knowledge of natural language [11] has revived with such a need that the efficiency of government institutions and production systems force to depend on it. It is no coincidence that among the software tools, the most popular are programs focused on processing natural language objects: text and linguistic editors and processors, programs for automatic correction of grammatical errors, automatic editing, natural language indexing, and search, as well as programs for machine translation, optical text recognition, etc. Recently, natural language modules have been increasingly included in the operating systems themselves.

All these problems are impossible without the involvement of a universal mathematical language [12]. Developments in this field have been carried out for several decades. Work was carried out on algebraization in logic, and a special mathematical apparatus was developed for the formula representation of relations and operations on them, which are called the algebra of finite predicates. The central place in the algebra of predicates is occupied by relations, which reflect the properties of objects and the connections between them. However, until now there is no convenient method of formulating the connection of free connections, which allows us to implement them programmatically. The possibility of a formula describing predictions or relationships is important when designing an automatic control system when developing a natural language intelligent interface.

As a result, the apparatus of logical spaces and the algebra of linear predicate operations were studied, and a formal representation of correspondence was developed using the algebraic apparatus of finite predicates, oriented to real calculations of the capabilities of the modern computer computing base and new requirements for information technologies, it is possible to abandon the modeling of any logical structures that require a lot of computation in real-time.

Although the primary focus is on modeling natural language structures, the method can be applied to any subject area. By analyzing the objects and their properties, appropriate semantic features and values are selected. This enables the method to explore complex data relationships, leading to deeper insights and improved decision-making, while ensuring robustness and broad applicability.

Objectives and approach

This study develops and formalizes a method based on the algebra of predicate equations for solving quantitative linear equations, while also reviewing the formal approaches to modeling intelligent systems. The proposed method overcomes the limitations of existing algebraic techniques—such as restricted scalability, difficulties in adapting to dynamic datasets, and inefficiencies in managing complex logical dependencies—by establishing a structured framework for representing logical relationships and constraints within the data. By expressing relationships within datasets through predicate equations, the approach formalizes dependencies and enhances analytical workflows. Designed to improve both accuracy and scalability while maintaining computational efficiency, it is ideally suited for practical data analysis applications.

To achieve these objectives, the research integrates theoretical and practical elements. The theoretical component formalizes the predicate algebra and explores its application in data analysis. On the practical side, algorithms are developed to demonstrate the method's effectiveness in tasks such as filtering, aggregation, and knowledge extraction, with benchmarks conducted to assess performance across datasets of varying sizes and complexity.

Section 1 details the method by defining the initial data—problem dimensions, semantic features, logical equations, and resulting variables—and describing an algorithm that verifies system consistency, deactivates non-essential features, and transforms a predicate equation into an equivalent operator (matrix) equation. Section 2 demonstrates the method’s practical application in a database context by modeling relationships (for example, between factories and parts) using predicate equations, which are then solved through corresponding operator equations to optimize query processing. Section 3 presents a detailed case study, complete with a flowchart and benchmark tests using synthetic datasets. The results confirm that the execution time scales linearly with the dataset size, thereby verifying the method’s efficiency and scalability. The final sections discuss the performance outcomes, suggest further optimizations (such as employing sparse matrices, parallel processing, or GPU acceleration), and summarize the contributions of this work to formalizing complex logical relationships and enhancing intelligent database systems.

1. Description of the method

At the beginning of the method is the data about the modeled system: the dimension of the problem, semantic features (input vector of data), a system of logical equations, and a set of resulting variables. After processing this data by the method, during which the consistency of the system of logical equations is determined and the non-essential features are turned off, we obtain the entire array of solutions of the system of logical equations at the output (Fig. 1).

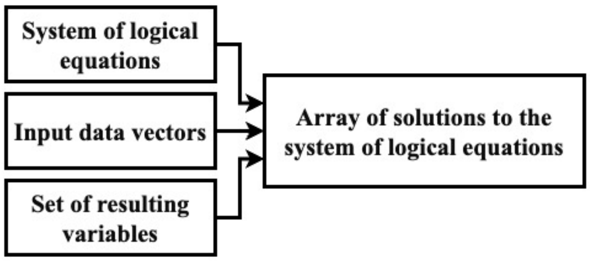
System of logical equations
Input data vectors
Set of resulting variables
Array of solutions to the system of logical equations

Fig. 1. Model of the developed system

Based on the theory of linear logical operators mentioned in the previous section, we will build an algorithm for solving the equation.

Let it be necessary to find a solution to the following predicate equation:

$$Q(Y) = \exists X(P(X) \vee TO(Y,X)), \tag{1}$$

where Q(Y) and P(X) — predicates are given on the branch $u=(u_1,...,u_n)$, consisting of n elements;
Y – vector;
X – predicate value;
TO(Y, X) — binary predicate is given on the branch $u \times u$.

It is necessary to calculate the predicate P(X), considering the predicate Q(Y) and the linear logical operator LO(Y, X) to be known.

Considering that the existence quantifier connects the predicate variable X, the equation can be rewritten in the form:

$$Q(Y) = \bigvee_{j=1}^{n}\left(P(u_j) \wedge K(Y,u_j)\right), \tag{2}$$

where Q, P — predicates;
Y – vector;
u – branch;
K – transposed matrix of the operator.

Equality is fulfilled only if it is true for any value of the predicate variable Y that runs through the set U. Thus, we have the following n equalities:

$$Q(u_i,u_j) = \bigvee_{i,j=1}^{n}\left(P(u_j) \wedge K(u_i,u_j)\right), \tag{3}$$

where Q, P — predicates;
u_i and u_j – branches;
K – transposed matrix of the operator.

$Q(u_i)$ and $P(u_j)$ – predicates by y_i and x_j $y_i, x_j \in \{0,1\}$ and $i,j \in 1,\ldots,n$. We denote the value of the binary predicate $TO(u_i, u_j)$ by $k_{ij} \in \{0, 1\}$, $i,j \in 1,\ldots,n$. Considering the following notations, equality will take the form

$$y_i = \bigvee_{j=0}^{n}\left(x_j \wedge k_{ij}\right), \tag{4}$$

where y_i and x_j — predicate values for any $i \in 1,...,n$ respectively yi, $x_j \in \{0, 1\}$ and $i,j \in 1,\ldots,n$;
k_{ij} – unary operator.

It is known that if for arbitrary predicates P(t) and Q(t) do the relation ψ: $P(t) \rightarrow X$. From here we get about the operator equation of the form:

$$LO(X) = Y, \tag{5}$$

where LO – linear logical operator;
X – predicate value;
Y – vector.

Thus, the predicate equation is equivalent to the operator equation, defined in the logical space. According to the idea of a continuous type of matrix of a linear-logical constant operator acting from space to itself, for reversibility, it is also necessary that in each row and column of the matrix of such an operator there should be one and only one element equal to one. If the matrix satisfies the above conditions of the idea, then the solution of the equation will be as follows

$$X = LO^{-1}(Y) \tag{6}$$

where LO – is a linear logical operator;
X – predicate value;
Y – vector.

The matrix of the inverse operator coincides with the transposed matrix of the operator K. Thus, the solution operator equation in the matrix type will be as follows:

$$X = RT * Y \tag{7}$$

where X – predicate value;
R – binary predicate;
T – logical vector;
Y – vector.

As a result of solving the predicate equation, it can be written in the form:

$$P(X) = \exists Y\left(Q(Y) \wedge LO(X,Y)\right), \quad (8)$$

where P(X) – predicate;
X – predicate value;
Q, Y – vectors;
LO – is a linear logical operator.

If the operator LO is not regular, the solution of the predicate equation cannot be written in the form, however, using the algebraic notation of the predicate equation, we will look for the solution of the equation in the following way.

Operator equation in the form of a system of logical equations. Assume that the vector Y is not singular.

$$\begin{cases} \bigvee_{j=1}^{n}(k_{mj} \wedge x_j) = y_m \\ \bigvee_{j=1}^{n}(k_{ij} \wedge x_j) = y_i \\ \bigvee_{j=1}^{n}(k_{nj} \wedge x_j) = y_n \end{cases} \quad (9)$$

where k_{ij} – unary operator;
y_i and x_j — predicate values for any $i \in 1,...,n$.

Let ones be worth in vector Y at places $(d_1,\ldots,d_y)=D$, and zeros at places $(z_1,\ldots,z_y)=Z$, $D \cap Z \neq \varnothing$, $D \cup Z=N$, $N=(1,...,n)$. The set of places where the zeros of the vector are X, we will denote as $L=(l_1,\ldots, l_x)$. The symbol * will mark the places where there can be zeros or ones. The algorithm is as follows:

1. Initialization $i:=z_1$;
2. We form the set consisting of zero coordinates of the vector X and j:=1. If TO[i,j] = 1, then X [j]:= l_1. We organize the sorting of indices j from 0 to n;
3. Index i is equated to the next element from the set Z and the transition to clause 2 until all the elements of the set Z are selected;
4. We form the set M. We obtain logical vector X consisting of zeros and symbols *;
5. Checking the system for consistency. Substituting the found vector into the system. We organize the solution of the system obtained according to the formula (13):

$$\bigvee_{j=1}^{n}\left(k_{ij} \wedge x_j\right) = y_i \quad (10)$$

where y_i and x_j — predicate values;
k_{ij} – unary operator.

If the system is incompatible, then the vector is not a solution to the system;

6. Formation of the system solutions. In vector $X^{(*)}$, we substitute a unit instead of the first symbol *, and zeros instead of the other symbols. Transition to clause 5. If the formed logical vector is a solution of the system, we store it in the array of solutions. We organize various substitutions of zeros and ones instead of * symbols, with each new combination going to item 5.
7. We write out all the solutions obtained from the system if the array of solutions is not empty. Furthermore, if not, the result is a message about the inconsistency of the system.

2. Example application

The given example in this section illustrates the possibility of using the theory of linear logical operators and the method of solving the quantifier predicate equation for processing and storing information in databases.

Suppose that the database contains information about four factories that produce parts for cars. Let factory z_1 produce parts d_1 and d_2, plant z_2, produces parts d_2 and d_3, factory z_3 produces parts d_1 and d_4, plant z_4 produces parts d_3 and d_4. The existing "plant-part" relationship is easily described by the following binary predicate.

As a result, the predicate $P_3(z)$ corresponding to the sought information is denoted by a quantifier equation of the form:

$$\exists d P_1(z,d) \wedge P_2(d) = P_3(z) \quad (11)$$

where d – parts;
P_1, P_2, P_3 – predicates;
z – factory;
d – parts.

The solution of this quantifier predicate equation is obtained from the solution of the corresponding operator equation A*X=Y in the linear logical space E^n. Matrix 15 represents the solution of the operator equation A*X=Y in the linear logical space E^n and the vector X=(1 0 0 0).

$$\begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{vmatrix} \quad (12)$$

Because of the action of operator A on vector X, we get the vector Y equal to (1 0 1 0) meaning that the parts d_1 are served by factories z_1 and z_3. Thus, the operation of searching for information of interest in the database is replaced by operating operator multiplication. Now it is necessary to calculate which factories produce parts d_1 or d_3. Using the additive property of the linear logical operator A, we obtain

$$A\times X_1 \vee A\times X_3 = A(X_1 \vee X_3) = A\times X_4, \qquad (13)$$

where A – linear logical operator;

X_1, X_2, X_4 – vectors created by predicates.

Vector X_1 and X_3 are created by predicates that describe the details of d_1 and d_3. Thus, the answers to more complex queries in the database also come from the solution of the operator equation. Using the algorithm for solving the operator equation, described in the previous subsection, it is possible to search for the parts that they manufacture by given factories. For example, let it be necessary to calculate which parts are manufactured by factory z_2. Therefore, the logical vector Y=(0 1 0 0).

$$\begin{vmatrix}1&1&0&0\\0&1&1&0\\0&0&0&1\\1&0&1&1\end{vmatrix} * \begin{vmatrix}x_1\\x_2\\x_3\\x_4\end{vmatrix} = \begin{vmatrix}0\\1\\0\\0\end{vmatrix} \qquad (14)$$

where x_1, x_2, x_3, x_4 – vectors.

Because of the decision of the operator equation of the formula 17.

Relative to X, we obtain the vectors (0 1 0 0) and (0 0 1 0). Therefore, factory z_2 produces parts d_2 and d_3.

Next, let's assume that the database contains information about which parts are used in certain machines. Let's assume that machine m_1 uses part d_2, Machine m_2 uses parts d_2 and d_3, Machine m_3 uses parts d_2 and d_3, and machine m_4 uses parts d_3 and d_4. This relationship "machine-part" corresponds to the binary predicate $K_1(m, d)$.

Similar to the previously considered case, it is easy to extract information about which machines and which parts are used from the database by solving the corresponding quantifier predicate equation, replacing it with an operator equation. We have the following equation:

$$\exists d K_1(m,d) \wedge K_2(m) = K_3(d), \qquad (15)$$

where d – parts;

K_1 – binary predicate;

$K_2(m)$ and $K_3(d)$ – unary predicates;

m – machines.

They specify a specific machine and a specific detail, respectively. Solving this equation concerning predicate $K_2(m)$ or predicate $K_3(d)$, we will pull the necessary information from the database. Suppose that now it is necessary to solve a more complex problem, namely: to calculate which factories produce parts for a predetermined machine. The following system of quantifier predicate equations corresponds to this condition:

$$\begin{cases}\exists d P_1(z,d) \wedge P_2(d) = P_3(z)\\ \exists d K_1(m,d) \wedge K_2(m) = P_2(d)\end{cases}, \qquad (16)$$

where d – parts;

K_1 – binary predicate;

K_2 – unary predicate;

P – predicates;

m – machines;

z – plant.

We will display this system in the form of one equation:

$$\exists d P_1(z,d) \wedge \left(\exists d K_1(m,d) \wedge K_2(m)\right) = P_3(z) \qquad (17)$$

where d – parts;

P_1 and P_3 – predicates;

z – plants;

K_1 – binary predicate;

K_2 – unary predicate;

m – machines.

The quantifier predicate equation corresponds to an operator equation of the form:

$$B\times T = X \qquad (18)$$

where B – operator matrix;

T – logical vector;

X – vectors created by predicates.

Logical vectors T and X are constructed, respectively, from the binary predicates $LO_2(m)$ and $LO_3(d)$. The linear logical operator U has the form matrix

$$\begin{vmatrix}0&1&0&0\\1&0&1&0\\0&1&1&1\\0&0&0&1\end{vmatrix} \qquad (19)$$

built on the binary predicate $TO_1(m, d)$. Therefore, the predicate equation corresponds to the operator equation of the form:

$$A(B(T)) = X, \qquad (20)$$

where A, B – operators matrix;
T – logical vector;
X – vectors created by predicates.

Let's transform the resulting equation into the following form:

$$Z(T) = X, \tag{21}$$

where Z – linear logical operator;
T – logical vector;
X – vectors created by predicates.

Z is equal to the superposition of operators A and B. In this case, we have:

$$C = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \end{vmatrix} * \begin{vmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix} \tag{22}$$

where C – operator matrix.

A rather large search in the database is reduced to the calculation of the matrix of the operator I_z using the operation of multiplying the matrices of the operators A and B. For example, let's calculate which factories make parts for machine m_1. The corresponding logical vector T has the following form (1 0 0 0). Thus, we have:

$$\begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix} * \begin{vmatrix} 1 \\ 0 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 0 \\ 0 \end{vmatrix} \tag{23}$$

As a result, we get that the manufacturers of machine parts m_1 there are plants z_1 and z_2.

3. Case study

This study provides a demonstration of the developed method for solving linear equations with quantifiers, aligned with the stated objectives, and implemented in Python to enable effective solutions. Integral to this approach, algebraic methods find numerous applications in databases [13, 14] and intelligent systems [15, 16], enhancing the efficiency of program translation and optimization. The algebra of queries formalizes request processing and information retrieval, making it essential for designing expert systems [17, 18]. This combined methodology facilitates a formal description of derived information based on basic information, optimizing query handling and retrieval processes.

In the context of this study, an important step is to translate query conditions into a system of linear equations, which allows for more efficient processing of database queries and information retrieval. Transforming selection and aggregation conditions into linear equations not only formalizes the logic of queries but also provides the ability to apply mathematical methods to data analysis.

To address the problem of identifying connections based on the example discussed earlier, we adopted a structured data model representing information as logical matrices. This approach allows us to express relationships using linear predicate equations, enabling efficient filtering, searching, and aggregation of data through matrix computations.

In the following example, we consider how specific queries can be transformed into a system of linear equations, illustrating this process in an abstract scenario.

Let's say we have a database data structure consisting of three main entities:

– Machines: each machine is uniquely identified by a MachineID and has a descriptive name (MachineName).

– Parts: Each part has a unique identifier (PartID) and a name (PartName).

– MachineParts (Machine-Part Relationships): This table defines the relationships between machines and parts, indicating which parts are used by each machine.

To process this data and perform logical operations efficiently, the relationships are transformed into a matrix M×P, where M is the number of machines (rows) and P is the number of parts (columns). In this matrix representation, each element A[i,j] is set to 1 if machine i uses part j, and 0 otherwise. This transformation enables the relationships to be expressed mathematically using linear predicate equations.

The core equation of this approach is represented as:

$$Y = A \cdot X \tag{24}$$

where A is the logical operator matrix derived from the MachineParts table, X is a binary vector representing the selection conditions for parts (X[j]=1 if part j is included in the query, 0 otherwise), and Y is the resulting binary vector where Y[i]=1 indicates that machine i satisfies the conditions imposed by X. For example, if a query specifies that we are interested in parts 101 and 103, the vector X would be represented as [1,0,1] for a dataset with three parts (101, 102, 103). Multiplying A by X will produce Y, identifying the machines that use at least one of the specified parts.

In addition to basic logical operations like "OR," more complex queries can be handled. For instance, finding machines that use both parts 101 and 103 involves a stricter condition, requiring Y[i] to equal 2 (indicating that machine i uses both parts). Similarly, negation ("NOT") can be applied by excluding specific parts from the selection vector X.

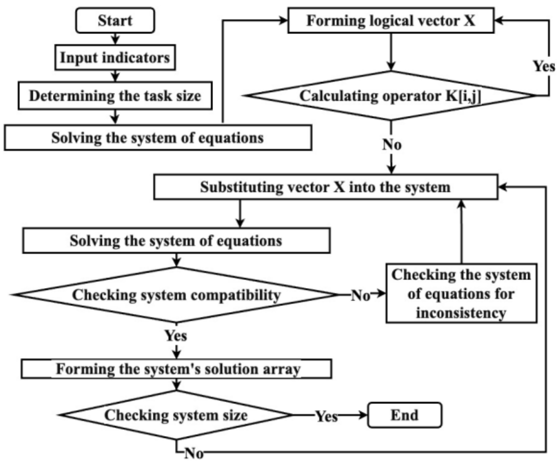
Start
Input indicators
Determining the task size
Solving the system of equations
Forming logical vector X
Calculating operator K[i,j]
Yes
No
Substituting vector X into the system
Solving the system of equations
Checking system compatibility
No
Checking the system of equations for inconsistency
Yes
Forming the system's solution array
Checking system size
Yes
End
No

Fig. 2. Flowchart of the method

To ensure a robust and scalable solution, the flowchart of the method outlines the logical process of solving such linear predicate equations (Fig. 2). This algorithm outlines the step-by-step process for solving a system of equations related to tasks in intelligent systems and databases.

The process begins with the input of necessary indicators and determining the size of the task. This step provides the foundation for subsequent calculations. Based on the input data, the logical vector X is formed, representing the initial data required for the system's operation. The next step involves calculating the operator K[i,j]. If the operator cannot be computed, the vector X is substituted into the system for further analysis.

If the system is compatible, the solution array is formed. If not, the algorithm performs an analysis to identify inconsistencies in the system. Once compatibility is confirmed, a solution array is generated, containing the computed values for the task. In the final stages, the system size is checked. If the task is complete, the algorithm terminates. Otherwise, it loops back to the specific steps for adjustments and further analysis. The algorithm concludes once a valid solution is found or the system is determined to be incompatible.

To validate the efficiency and scalability of the proposed method, an updated benchmark was conducted using synthetic datasets of increasing size.

The benchmark was performed on a machine with an Intel i7 processor and 16 GB of RAM.

– Software: Python 3.9, NumPy for matrix operations.

– Datasets: Synthetic datasets were generated with the following:

– Machine counts: 100, 500, 1,000, 5,000, and 10,000.

– Part counts: 1,000, 5,000, 10,000, 50,000, and 100,000.

– Relationship density: 10% (to simulate real-world sparsity).

For each dataset size, the function find_machines_with_parts was run with:

– A random selection of 10 parts to include.

– A random selection of 5 parts to exclude.

Execution time was measured for each configuration and averaged over three runs.

Table 1 summarizes the updated benchmark results.

Table 1

Number of Machines	**Number of Parts**	**Execution Time (seconds)**
100	1000	0.0000~
500	5000	0.0028
1000	10000	0.0113
5000	50000	0.2877
10000	100000	1.1573

The execution time scales predictably with the size of the dataset, demonstrating linear growth. The results confirm the method's ability to handle large datasets effectively.

The execution times are plotted below in Figure 3, showing how the time required increases linearly with the number of machines.

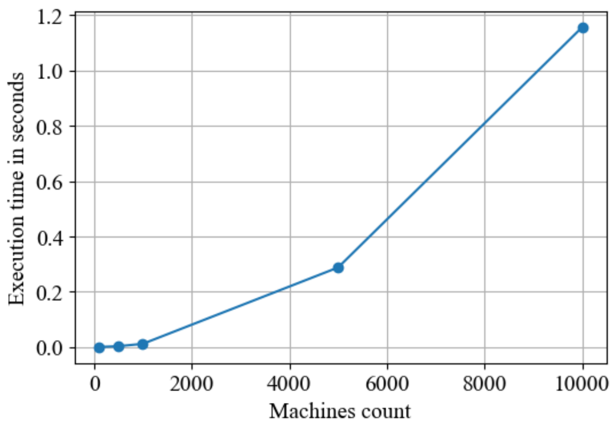

Fig. 3. Execution times

Performance results:

– For small datasets (e.g., 100 machines, 1,000 parts), the execution time is negligible (0.0000 seconds).

– For larger datasets (e.g., 10,000 machines, 100,000 parts), the method maintains reasonable performance, processing the query in just over one second.

These results indicate that the method can handle real-world datasets of medium to large sizes efficiently. For example, processing data with 10,000 machines and 100,000 parts representing a realistic industrial scenario requires just over a second.

4. Discussion

The benchmark results demonstrate that the proposed method for solving linear predicate equations is both efficient and scalable. The key observations from the benchmark analysis are summarized below, highlighting the performance, scalability, and potential areas for further improvement.

The benchmark shows that the method scales linearly with the size of the dataset, as evidenced by the nearly proportional increase in execution time concerning the number of machines and parts. This linear complexity arises from the matrix-vector multiplication at the core of the method, which is computationally efficient. For datasets with up to 10,000 machines and 100,000 parts, the execution time remains just over one second, making the approach practical for medium to large-scale real-world applications.

The results confirm the method's suitability for real-world use cases. Scenarios involving medium to large datasets, such as industrial systems with thousands of machines and parts, can be handled with minimal computational overhead. This makes the method ideal for applications in intelligent systems, database optimization, and information retrieval.

While the method performs well computationally, memory usage could become a limiting factor for extremely large datasets with millions of machines and parts. This is particularly relevant for dense datasets where the matrix representation might require significant storage.

Future Optimization Opportunities:

1. Sparse Matrix Optimization: For datasets where the relationships between machines and parts are sparse (a common scenario in industrial applications), leveraging sparse matrix libraries, such as scipy.sparse, could significantly reduce memory usage and improve computational efficiency.

2. Parallel and Distributed Computing: The method could benefit from parallelization, especially for datasets with millions of rows and columns. Using multi-threading, multiprocessing, or distributed computing frameworks could further improve performance.

3. GPU Acceleration: By leveraging GPU-based matrix computations using libraries such as cupy or PyTorch, the method could achieve substantial speedups for very large-scale datasets.

The proposed method provides a scalable and efficient solution for solving linear predicate equations, demonstrating robust performance on datasets of various sizes. The linear growth in execution time and practical applicability to large datasets make it a valuable tool for a range of applications. However, for datasets exceeding millions of entries, additional optimizations such as sparse matrix representations, parallelization, or GPU acceleration would be necessary to maintain the method's efficiency and scalability. These enhancements could further extend its applicability to massive datasets in modern intelligent systems and data-driven industries.

Conclusions

The presented work introduces a novel approach for the formal representation of complex relationships and operations in intelligent systems through the algebra of finite predicates. This approach focuses on addressing the lack of a convenient method for programmatically implementing formulaic connections of free relationships, which is critical for applications in natural language processing and automatic control systems.

This research contributes to the field by developing a formal representation of correspondence using logical spaces and the algebra of linear predicate operations. This representation is tailored for real-time computations, leveraging the capabilities of modern computing technologies and addressing the new demands of information systems. The novelty lies in the flexibility of the proposed method, which allows the modeling of diverse subject areas by systematically defining and using semantic features.

From a practical perspective, the proposed method facilitates the development of intelligent interfaces, such as natural language processing systems, which are crucial for applications like machine translation, grammatical error correction, and optical text recognition. Moreover, the approach ensures scalability and robustness, enabling its application across various domains, from knowledge systems [19] to decision-making frameworks, ultimately enhancing the efficiency and adaptability of intelligent systems [20].

Contributions of authors: conceptualization, methodology, formulation of tasks – **Zoya Dudar;** analysis – **Volodymyr Liashyk;** development of model, verification – **Volodymyr Liashyk;** analysis of results, visualization – **Zoya Dudar;** writing – original draft preparation visualization – **Volodymyr Liashyk;** writing – review and editing – **Zoya Dudar.**

Conflict of Interest

The authors declare that they have no conflict of interest about this research, whether financial, personal, authorship or otherwise, that could affect the research, and its results presented in this paper.

Financing

This study was conducted without financial support.

Data Availability

The work has associated data in the data repository.

Use of Artificial Intelligence

The authors confirm they did not use artificial intelligence methods while creating the presented work.

All authors have read and agreed to the published version of this manuscript.

References

1. Karataiev, O., & Shubin, I. Formal Model of Multi-Agent Architecture of a Software System Based on Knowledge Interpretation. *Radioelectronic and Computer Systems*, 2023, no. 4, pp. 53–64. DOI: 10.32620/reks.2023.4.05.
2. Shubin, I., Kozyriev, A., Liashyk, V., & Chetverykov, G Methods of adaptive knowledge testing based on the theory of logical networks. *CEUR Workshop Proceedings*, 2021, vol. 2870, pp. 1184-1193. Available at: https://ceur-ws.org/Vol-2870/paper86.pdf (accessed May 11, 2024).
3. Shubin, I. Development of Conjunctive Decomposition Tools. *CEUR Workshop Proceedings*, 2021, vol. 2870, pp. 890-900. Available at: https://ceur-ws.org/Vol-2870/paper67.pdf (accessed May 15, 2024).
4. Jansma, A., Mediano, P. A. M., & Rosas, F. E. *The Fast Möbius Transform: An algebraic approach to information decomposition*, 2024. DOI: 10.48550/arXiv.2410.06224.
5. Dudar, Z., & Litvin, S. *Metod ontolohichnoho opysu v pobudovi servis-oriyentovanykh system rozpodilenoho navchannya* [Formalization and Application of Algebraic Methods in Automated Intelligent Systems]. *Suchasnyy stan naukovykh doslidzhen' ta tekhnolohiy v promyslovosti — Innovative Technologies and Scientific Solutions for Industries*, 2024, pp. 39-53. DOI: 10.30837/ITSSI.2024.27.039. (In Ukrainian).
6. Zhanlav, T., Otgondorj, Kh., Mijiddorj, R.-O., & Saruul, L. *A unified approach to the construction of higher-order derivative-free iterative methods for solving systems of nonlinear equations*. Proceedings of the Mongolian Academy of Sciences, 2024, vol. 64, no. 02, pp. 24–35. DOI: 10.5564/pmas.v64i02.3649.
7. Shubin, I., & Karataiev, O. *Problemy povtornoho vykorystannya znan'u protsesi proyektuvannya prohramnykh system* [Reuse of information based on the interpretation of knowledge]. *Suchasnyy stan naukovykh doslidzhen' ta tekhnolohiy v promyslovosti – Innovative Technologies and Scientific Solutions for Industries*, 2023, no. 2, pp. 62-71. DOI: 10.30837/ITSSI.2023.24.062. (In Ukrainian).
8. Omran, P. G, Wang, Z., & Wang, K., Learning Rules with Attributes and Relations in Knowledge Graphs. *AAAI Spring Symposium: MAKE*, 2020, vol. 3121. Available at: https://ceur-ws.org/Vol-3121/paper10.pdf (accessed 2 April 2024).
9. Vysotska V., Shubin I., Mezentsev M., Kobernyk K., & Chetverikov G Ukrainian Big Data: The Problem Of Databases Localization. *The 8th International Conference on Computational Linguistics and Intelligent Systems. (COLINS-2024)*, Lviv, Ukraine, IEEE, April 12–13, 2024, vol. 3688, pp. 122–133. Available at: https://ceur-ws.org/Vol-3688/paper9.pdf (accessed May 14, 2024).
10. Pellissier-Tanon, T., Weikum, G., & Suchanek, F., YAGO 4: A Reason-able Knowledge Base. *17th International Conference, ESWC 2020*, Heraklion, Crete, Greece, IEEE, 2020, pp. 583-596. DOI: 10.1007/978-3-030-49461-2_34.
11. Barkovska, O. Research into Speech-to-text Transformation Module in the Proposed Model of a Speaker's Automatic Speech Annotation. *Innovative Technologies and Scientific Solutions for Industries*, 2022, no. 4, vol. 22, pp. 5-13. DOI: 10.30837/ITSSI.2022.22.005.
12. Martinsson, A., & Su, P. Mastermind with a Linear Number of Queries. *Journal of Mathematical Analysis and Applications*, 2023, no. 3, pp. 92-94. DOI: 10.48550/arXiv.2011.05921.
13. Kamide, N. Sequential Fuzzy Description Logic. Reasoning for Fuzzy Knowledge Bases with Sequential Information. *2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL)*, Miyazaki, Japan, IEEE, 2020 pp. 218–223. DOI: 10.1109/ISMVL49045.2020.000-2.
14. Qing-Hu, H., & Yarong, W. Rational solutions to the first order difference equations in the bivariate difference field. *Journal of Symbolic Computation*, 2024, vol. 124. DOI: 10.1016/j.jsc.2024.102308.
15. Shivappriya, S. N., Priyadarsini, M. J. P., Stateczny, A., Puttamadappa, C., & Parameshachari, B. D. Cascade Object Detection and Remote Sensing Object Detection Method Based on Trainable Activation Function. *Remote Sensing*, 2021, vol. 13, no. 2. DOI: 10.3390/rs13020200.
16. Zhao, Y., & Tao, C. The accurate and efficient solutions of linear systems for generalized sign regular matrices with certain signature. *Journal of Computational and Applied Mathematics*, 2023, vol. 431, article no. 115280. DOI: 10.1016/j.cam.2023.115280.
17. Shubin, I., Snisar, S., & Litvin, S. Categorical Analysis of Logical Networks in Application to

Intelligent Radar Systems. *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC S&T)*, 6–9 October 2020, Kharkiv, Ukraine, IEEE, pp. 235-238. DOI: 10.1109/picst51311.2020.9467893.

18. Karataiev A., & Shubin I. *Formalna model multyahentnoi arkhitektury prohramnoi systemy na osnovi interpretatsii znan* [Formal model of multi-agent architecture of a software system based on knowledge interpretation]. *Radioelectronic and Computer Systems*, 2023, no. 4, pp. 53–64. DOI: 10.32620/reks.2023.4.05.

19. Chen, Z., & Wang, Y. Knowledge graph completion: A review. *IEEE Access*, 2020, vol. 8, pp. 192435–192456. DOI: 10.1109/ACCESS.2020. 3030076.

20. Beskorovainyi, V., Kuropatenko, O., & Gobov, D. Optimization of transportation routes in a closed logistics system. *Innovative Technologies and Scientific Solutions for Industries*, 2019, pp. 24-32. DOI: 10.30837/2522-9818.2019.10.024.

Received 17.10.2024, Accepted 17.02.2025

МЕТОД РОЗВ'ЯЗАННЯ КВАНТОРНИХ ЛІНІЙНИХ РІВНЯНЬ НА БАЗІ АЛГЕБРИ ЛІНІЙНИХ ПРЕДИКАТНИХ ОПЕРАЦІЙ

З. В. Дудар, В. А. Ляшик

Предметом статті є підходи, які розширюють існуючий набір математичних інструментів для обробки складних зв'язків у базах даних і обчислювальних системах. Це особливо важливо для додатків, які вимагають ефективного пошуку інформації, представлення знань і логічного висновку в автоматизованих середовищах прийняття рішень. **Завдання** передбачає розробку методу розв'язання кванторних лінійних рівнянь з використанням алгебри лінійних предикатних операцій, спрямованого на покращення оптимізації запитів до бази даних та розширення можливостей інтелектуальних систем. **Методи**, які використовуються в цьому дослідженні, включають алгебраїчні методи, логічні операції та матричні перетворення для моделювання та ефективного розв'язання предикатних рівнянь. Використовуючи алгебру кінцевих предикатів, запропонований підхід забезпечує більш систематичний і масштабований спосіб обробки логічних залежностей і оптимізації обчислювальних процесів. Метод інтегрує лінійні логічні оператори, гарантуючи, що складні запити та обмеження в базах даних можуть бути представлені та оброблені за допомогою формальних математичних моделей. Крім того, він представляє структуру, яка покращує структурне представлення знань, полегшуючи інтелектуальний аналіз даних. У **результаті** дослідження було розроблено формальний метод розв'язання кванторних лінійних рівнянь, що забезпечує більш ефективну оптимізацію запитів, логічне мислення та механізми підтримки прийняття рішень в експертних та автоматизованих інформаційних системах. Дослідження демонструє, що алгебраїчні підходи можуть значно підвищити ефективність процесів пошуку інформації, особливо в інтелектуальних базах даних, де реляційні обмеження та залежності відіграють вирішальну роль. Тестування, проведене на синтетичних наборах даних, підтверджує масштабованість методу, показуючи, що він підтримує лінійне зростання часу виконання навіть із збільшенням складності даних. **Висновок**: запропонований метод розширює математичну основу для вирішення логічних рівнянь в обчислювальних середовищах, забезпечуючи потужний інструмент для інтелектуальних систем і оптимізації баз даних. Здатність формалізувати та обробляти складні логічні зв'язки сприяє підвищенню точності прийняття рішень та ефективності автоматизації.

Ключові слова: інтелектуальні системи; алгебраїчні методи; формальне моделювання; логічні методи; алгебра скінченних предикатів; предикатне рівняння.

Дудар Зоя Володимирівна – канд. техн. наук, проф., зав. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

Ляшик Володимир Андрійович – асп. каф. програмної інженерії, Харківський національний університет радіоелектроніки, Харків, Україна.

Zoya Dudar – Candidate of Technical Sciences, Professor, Head of the Department of Software Engineering, Kharkiv National University of Radioelectronics, Kharkiv, Ukraine,
e-mail: zoya.dudar@nure.ua, ORCID: 0000-0001-5728-9253, Scopus Author ID: 6506991522.

Volodymyr Liashyk – Postgraduate Student of the Department of Software Engineering, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine,
e-mail: volodymyr.liashyk@nure.ua, ORCID: 0000-0001-7326-0813.