Method for Solving Quantifier Linear Equations for Formation of Optimal Queries to Databases

Igor Shubin, Andrii Kozyriev

Kharkiv National University of Radio Electronics, Nauky ave. 14, Kharkiv, 61166, Ukraine

Abstract

This paper is devoted to the study and development of a logical model that implements the method of solving quantifier linear equations. The model created on the basis of the theory of linear logical operators and the method of solving the quantifier predicate equation, can be used to solve the problem of logical results in databases, that is, to process and store information in databases, as well as to create natural language interfaces in computer systems.

Keywords

Equations, quantifier linear equations, logical equations, logical methods, predicates, predicate algebra, predicate operations, databases, database queries, knowledge bases.

1. Introduction

Widespread use of computer technology and its rapid development led to high rates of development of methods for creating intelligent systems (IS) for various purposes. This led to the expansion of the range of problems solved by computers, to the increase of their role in human life. However, all this progress is purely quantitative in nature. Simple expansion of the computer's functionality is effective only if people are able to maintain it, and if not, then such expansion becomes pointless. At this time, methodological and technical approaches to the creation and use of information systems have already been developed. Currently available intelligent information systems are able to perform functions that were previously considered the exclusive prerogative of humans: prove mathematical theorems, translate texts from one language to another, diagnose diseases and perform many other functions. However, in the future, an ideal computing machine should surpass the human ability to think logically, analyze the information received, solve the most complex problems, and interact with the environment.

Relational and logical methods of knowledge representation play an important role in the development of mathematical support for information systems. One of the effective universal mathematical methods for describing information is the algebra of predicates and predicate operations. This language of algebra is easy and convenient to describe various formalized information, form queries in databases and model human activity [1].

In various computerized industries, there is a need to process information displayed in natural language. In automated computer system (ACS), which includes a person as its organic link, the main form of information transfer is served by documents containing a significant amount of textual information. Computer modeling of text processing will allow automating many types of human intellectual activity, expanding its capabilities. The basis of ACS is automated information systems (AIS), the purpose of which is to automate the processes of information accumulation, search, and generalization. The effectiveness of AIS is recognized by their ability to process unformalized or weakly formalized information.

COLINS-2023: 7th International Conference on Computational Linguistics and Intelligent Systems, April 20–21, 2023, Kharkiv, Ukraine
EMAIL: igor.shubin@nure.com (I. Shubin); andrii.kozyriev@nure.ua (A. Kozyriev)
ORCID: 0000-0002-1073-023X (I. Shubin); 0000-0001-6383-5222 (A. Kozyriev)

The goal of the work is the algorithmic implementation of the method of solving quantifier linear equations based on the algebra of linear predicate operations, the formal apparatus of linear logical operators and methods of solving logical equations.

2. Analysis of the subject field and statement of the problem

The form of presentation of information in the computer systems has a great influence on the speed and quality of information processing by intelligent systems [2]. Generally speaking, different information systems use different ways of presenting knowledge depending on the specific fields of application of the systems.

Representation of knowledge is a formalization of inherent beliefs with the help of figures, records or languages. Of special interest are the formalizations perceived by the computer. In this regard, formal languages are being developed that allow displaying knowledge in computer memory.

A characteristic feature of the operation of modern intelligent systems is that the data processing is based on the necessary knowledge of the problem industry, entered in advance in the knowledge base of the system, while previously created systems worked with data that were simply processed by various programs. A rather huge number of works have been devoted to the study of the difference between data and knowledge, the main idea of which can be formulated as follows: knowledge is a complexly organized type of data that differs from traditional ideas about data in four main features:

- Knowledge placed in the record contains not only the information part, but also the descriptive part – it stores all the data about the information unit that may be needed in the user's work with the system
- Knowledge in the knowledge base creates complex hierarchical structures, which is achieved by introducing various relations between information units entered in the knowledge base
- Informational units reflecting knowledge can be composed into more complex organized units and decomposed into simpler ones
- Attached or built-in procedures can act as parts of information units characterizing some knowledge, which allows these procedures to be activated as a result of the appearance of various information units or connections between them in the knowledge base

The peculiarity of systems of knowledge representations is that they model human activity, which is often carried out in an atypical variety. Thus, an important stage in the development of intelligent systems is the creation of an optimal model of representations of knowledge about the subject branch of the systems application [3]. It is obvious that the choice of a certain type of knowledge representation depends on the fields of formalization. In recent years, a large number of different models of knowledge representation have been proposed.

However, among the different ways of presenting knowledge, aroused from the specificity of the variety of knowledge structures, it is possible to single out a logical model, frame and production system, and semantic grids. Each method of presentation has its advantages and disadvantages, and is associated with a certain structure and fields of application of knowledge.

The logical model of knowledge representation uses the logic of first order predicates and the results derivation using the syllogism method. The predicate difference used in the logical model can be easily combined with a fairly effective result mechanism, such as a resolution. The advantages of logical models are the unity of the theoretical justification and the possibility of implementing a system of formally precise definitions and results. This is the reason why intelligent systems using a logical model of knowledge have become quite widespread.

The disadvantages of using the logical construction of systems include unconstructiveness and semantic limitations. At the same time, human logic is often not limited by the usual formalism of logical languages and is an intellectual model with a vague structure. However, the framework of formal logic is expanding more, which is the reason of appearance of modal, multi-valued and probabilistic logics. This makes it possible to expand the possibilities of applying logic in information systems.

When the volume of knowledge increases, it is reasonable to apply various methods of preliminary grouping and structuring of knowledge. The production system can be applied with a frame model, which gives good results.

In production models, knowledge appears to be a series of "how-to" rules. Such systems can produce direct or reverse results. The carriers of these models are the MYCIN system, designed to solve problems of a diagnostic nature, and the OPS system, designed to solve design problems.

Advantages of production systems are:

- Simplicity of creation and understanding of the rules
- Simplicity of replenishment and modification
- Ease of implementation of the mechanism of a logical result

The disadvantages of such systems include:

- Ambiguity of mutual differences between the rules
- Comprehension of the wholesome improvement of knowledge
- Difference from the human structure of knowledge
- Flexibility in a logical result

2.1. Overview of application branches of algebraic methods in automated systems

The recent practical additions to modern abstract algebra in databases and intelligent systems have led to increased interest in the possibility of algebraic description of information. At the same time, practice suggests unexpected new structures that enrich algebra. Based on the application of algebraic methods in programming theory, various translators from high-level languages and various algorithmic algebras were developed.

Automation of the development of software systems and computer design is an important and urgent problem of computer technology, which requires the development of a utilitarian theory of algorithms. One of the main problems of this theory is the problem of an optimal translator from one language to another, which is described in the next problem: there are two algorithmic languages and some algorithm implemented in one of them. It is necessary to find the optimal implementation of this algorithm in another language according to the given criteria. When performing applied tasks, as a rule, the first language is a high-level language focused on a certain range of tasks, and the second is the internal language of the machine [4].

Thus, it is necessary to translate from programming language to machine language with simultaneous optimization of the source program. The process of solving such a problem is divided into a number of intermediate stages, at each of which a partial optimization of the algorithm and translation into an intermediate language corresponding to this stage is carried out.

Say $S = \{f\}$ – the set of languages, and $P = \left\{ \overset{m}{}_{f', f'', f} \right.$ a set of translators, each representing a program in the language f and translates programs from the input language f' to output language f''. The translator can be considered as a unary operation, with a branch of definition represented by the language f' and a field of values f''.

We can define translators (1)

$$\left\{ \overset{m^1}{}_{f', f'', f_1} \text{ and } \right. \left\{ \overset{m^2}{}_{f_1', f_2'', f} \right. \tag{1}$$

And the transcoding operation (2) as follows

$$\overset{m^2}{}_{f_1, f_2, f} \left[\overset{m^1}{}_{f', f'', f_1} \right] = \overset{m^1}{}_{f', f'', f_2} \tag{2}$$

A composition operation (3) takes place over translators $\overset{m}{}_{f_1, f_2, f_1}$ and $\overset{m}{}_{f_2, f_3, f}$ as follows

$$\overset{m}{}_{f_1, f_2, f_1} \times \overset{m}{}_{f_2, f_3, f} = \overset{m}{}_{f_1, f_3, f} \tag{3}$$

The following operations can be used to formalize processes often used in programming.

To implement equivalent transformations of algorithms, it is necessary to build an algebra of algorithms that would allow to do transformations using a clear algebraic language.

We consider a database as an information system that stores and processes information and is able to provide answers to queries. Moreover, it should be possible to obtain not only information directly

stored in the database, but also derivative information obtained on the basis of basic information. The task of obtaining derivative information is directly related to the task of the result in intelligent systems.

In the case of applying an algebraic approach to the description of derived information, a certain algebraic system is distinguished – the algebra of queries, in terms of which the derived information is written through the basic one. We will display the database in the form of some mathematical model. Suppose that the system of a set of data – domains is represented as $D = (di, i \in F)$, where F is the set of domains.

For each set system $D = (di, i \in F)$ the symbol $\varphi \in \phi$ of type τ is implemented as a subset of the relation in the Cartesian multiplication $D_{i_1} \times \ldots \times D_{i_n}$, where type τ characterizes placing of variables. Say $n: X \to F$ which splits set X into categories: $X = X_1 \cup X_2 \ldots X_K$, where K is cardinality of sets F. The set X has a sufficient number of variables to implement different queries. We will consider the system of indicators $D = (di, i \in F)$ and the relations entered on it as a model. The rule by which every symbol φ of the relation ϕ is realized is denoted by f. Thus, the position of the database in the considered scheme with a given data system can be interpreted as a function f that compares each $\varphi \in \phi$ of type τ which is some subset of $D_{i_1} \times \ldots \times D_{i_n}$. Defining such model of database as (D, ϕ, F) the symbols of relations to be implemented in certain moments.

Next, it is necessary to implement the possibility of a query in the database. Query U in the state f defined as $f * z$. The corresponding query responses are given as a subset of D. The set $f * z$ is defined for the base queries U, and an arbitrary query is expressed in terms of the base queries.

Define the set of all subsets of D as MD, and the set of all possible queries as U. For each query U of U and each position f, the answer to the query is an element in MD. In order for an arbitrary query to be expressed in any form through basic queries, it is necessary to include algebraic operations on the set of queries U that allow operating with queries. Likewise, similar algebraic operations should be introduced on the set of answers. In this case, using samples of existing algebraic operations, the answer to an arbitrary query can be calculated in accordance with the structure of the query, written down through queries, the answers to which are already known. The sets of requests and responses considered below are algebras of requests and responses.

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 [5]. There are all sorts of differences between classical and non-classical logic. Boolean algebras, for example, correspond to the classical difference of statements, and special Heiting algebras correspond to the intuitionistic difference of statements.

2.2. Review of logic algebraization methods

At different times, various algebraic structures were introduced, related to the difference in order usually connecting cylindrical Tarsky algebras and polyadic Halmosh algebras. A Halmosh algebra is denoted by adding operations to a quantifier algebra, which in turn is a Boolean algebra with certain additional quantifier operations given on this algebra. Taking the quantifier can be represented in the form of some operation defined in the corresponding algebra. Next, various ways of defining such operations are considered, but first we will study the geometric meaning of quantifiers as operations.

Say the variables x and v given on the set U. Suppose that B is a subset of the set U, which is the Cartesian product of the set U times itself. The set U can be interpreted as a binary predicate $B(x, y)$, defined on U^2 and equal to one on all pairs $(x, y) \in B$. Say sets UX and UY as the corresponding projections of the set U^2. According to the definition of the existence quantifier, the expression $\exists\, x\, B(x, y)$ means that a unary predicate from variable y is defined on the set U, which defines some subset in the set UY consisting of elements $B_Y \in UY$, for which there exist $B_X \in UX$ such that $(B_X, B_Y) \in B$. Thus, the application of the existence quantifier to the binary predicate $B(x, y)$ gives a unary predicate on the variable B on the set Uv, which determines the set of elements of BX. Geometrically, $\exists\, x\, B(x, y)$ is the projection of the set B onto UY.

Likewise, the application of the existence quantifier by the variable y to the predicate $\mathrm{B}(x, y)$ determines the projection of the set B onto UY. The quantifier of generality is denoted by a dual form. Geometrically, $\exists\, x\, B(x, y)$ is the projection of the largest cylinder lying in Y on the set UY.

In the case of multiple relations, the application of the existence quantifier by k variables ($k < n$), where n is the local predicate $P(x_1, \dots, x_n)$, given on the set $U_n = U_1 \times \dots \times U_n$, can be interpreted as the projection of some subset of the set U_n on the set $U^{n-k} = \overset{j=1}{\overset{n-k}{\times}} U_j^n$. Thus, on the set U^{n-k}, an $n-k$ local predicate is defined by the application of the existence quantifier on the k variables of the predicate $P(\mathrm{x}_1, \dots, x_n)$. This expression is equivalent to the following equation (4):

$$\exists x_{i_1}, \dots, x_{i_k}\ P(x_1, \dots x_n) = Q(x_{i_{n-k}}, \dots, x_n) \tag{4}$$

When $k = n - 1$, this equality is a partial case of a linear disjunction operator with respect to the operation (5)

$$\exists x_1, \dots, x_{n-1}\ P(x_1, \dots x_n) \wedge R_1(x_1) \wedge \dots \wedge R_{n-1}(x_{n-1}) = Q(x_n) \quad \text{when } R_1(x_1) = 1, \dots, R_{n-1}(x_{n-1}) = 1 \tag{5}$$

Thus, the existence quantifier can be calculated as an operator linear with respect to the disjunction operation with the additional condition (5). Accordingly, the generality quantifier is denoted by its dual form as a logical operator linear with respect to the conjunction operation with an additional condition.

Defining H as a Boolean algebra. The quantifier of the existence of this algebra is called an arbitrary mapping $\exists: H \to H$, which satisfies these conditions:

1. $0 = 0$
2. $a > \exists a$
3. $\exists(\mathrm{a}_1 \wedge \exists \mathrm{a}_2) = \exists \mathrm{a}_1 \wedge \exists a_2, \cdots a, a_1, a_2 \in H$

The quantifier of generality is denoted by a dual form and is related to the quantifier of existence by means of the negation operation $\forall a = \overline{\exists \overline{a}}$.

However, the difference of first-order predicates is characterized by essential limitations, due to the fact that the regularity of first-order predicates, contains "theoretically inviolable frameworks of description, which are unconditionally delineated and strictly limited." [6] The requirements proposed for modern information systems make it necessary to spread the descriptive capabilities of the logical languages used by them. It is possible to extend the syntactic rules of the first-order predicate logic language to allow the use of mutable predicate symbols. As a result of this extension of language syntax, we have obtained a system called second-order predicate logic. The constructed system can include as arguments of predicates not only terms, but also predicate sentences of the first order. It is obvious that such a system of differences has much greater descriptive possibilities, why the logic of predicates of the first order.

From the difference of predicates of the second order, it is possible to connect the algebra of predicates and predicate operations, described in work [7]. Thus, depending on which difference to base the databases or knowledge bases discussed earlier, it is necessary to proceed from advantageous query and response algebras. In each case, the database in the main approximation can be considered as a automaton of type (6)

$$(L, U_{query}, U_{res}) \tag{6}$$

where L is the set of automaton states, U_{query} is the set of queries, U_{res} is the set of results. Next is defined the operation $L \times U_{query} - U_{res}$ the meaning of which lies in the second. If $1 \in L$ – the state of database, $U_{query} \in U_{query}$ – some query, then $1 * U_{query} = U_{res}$ is the answer to the noted request U_{res} in the proper state of the database.

Representation of the database in the form of an algebraic structure is shown to be useful in various cases. For example, in cases where it is necessary to determine the deviation of compositions and decompositions of databases, as well as from isomorphism and equivalence.

In the theory of databases, the relational data model has gained wide application. Based on the relational model of databases, it is possible to show the advantages of the algebraic approach to information description [8]. They were shown how to formulate relational database queries in relational algebra languages. The operations of selection, projection, set-theoretic union and conjugation were introduced as operations of relational algebra. The main advantage of relational algebra is that it is closed with respect to all relational operations. This means that the result of any operation is a new relation that has exactly the same status as the original one, in the sense that all algebraic operations are applicable to the resulting relation. No algebraic operation can create an object that goes beyond algebra, while in languages based on the difference, in order to formulate some complex queries, it is

necessary to formulate auxiliary subqueries, to create new constructions. With applied algebra, there is no need to create new structures when describing new relations, so it is possible to create queries of any complexity.

When designing relational databases, the knowledge about the subject area is presented in the form of relations of some groundless arity, this way of demonstrating knowledge is also effective when designing expert systems for various purposes [9]. Each relation can be mutually uniquely assigned a finite predicate, which, in turn, is encoded by a sequence of zeros and ones. Thus, a transition from relations on finite sets to binary codes of finite length is possible.

The information that remains in a fairly large set of binary codes suggests the existence of significant logical dependencies between the codes. Thus, some codes can be expressed through each other using logical operations, which indicates the necessity of the available information. The analysis of such dependencies requires the development of effective mathematical methods that allow describing the relationship between codes in the correct logical-algebraic language.

As operations in the algebra of binary codes, the functions of the algebra of logic applied to the elements of the algebra of bitwise codes. The concept of base codes is also used [10], which are the elements that can be used to obtain an arbitrary code by applying various available operations. Irreducible with respect to some operations of the system of binary codes are considered, when no element of the system can be obtained by applying to other superpositions of these operations. Based on a number of heuristic ideas about information processing, among a large class of all kinds of transformations of information represented by binary codes, it is natural to single out a class of transformations of codes that are linear with respect to disjunction or conjunction.

When a transformation linear with respect to the disjunction operation, defined on a set of binary codes, there is the operator A that transfers one code to another and satisfies the following two conditions (7)

$$\begin{cases} A(0) = 0 \\ A(X \vee Y) = A(X) \vee A(Y) \end{cases} \quad (7)$$

where $0 = (0, \dots, 0)$, $X = (x_1, \dots, x_n)$. Accordingly, an operator that is linear with respect to a conjunction is denoted by a dual form. The corresponding conditions are written in the second form (8)

$$\begin{cases} A(1) = 0 \\ A(X \wedge Y) = A(X) \wedge A(Y) \end{cases} \quad (8)$$

As a result of adding linear operators to the already existing operations of the algebra of binary codes, we obtained an algebraic system that possesses a number of fascinating properties. The resulting algebra, along with the disjunctive, conjunctive algebra, and the algebra of propositional operations, makes it possible to use a convenient algebraic language to formally write down the conditions that the described system of relations must satisfy. In particular, it is convenient when the information represented by binary codes has properties of linearity and homogeneity.

3. Description of the object model, methods and algorithms

3.1. The model of solving a quantifier linear equation

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

Suppose the problem is to find a solution to the following predicate equation (9)

$$Q(Y) = \exists X \, (P(X) \wedge K(Y, X)) \quad (9)$$

The predicates $Q(Y)$ and $P(X)$ are defined on the set $U = (U_1, \dots, U_n)$, which consists of n elements, and the binary predicate $K(Y, X)$, which is defined on the set $U \times U$. The problem is to calculate the predicate $P(X)$, considering the predicates $Q(Y)$ and $K(Y, X)$ are to be known.

Considering that the predicate variable X is connected by the existence quantifier, equation (9) will be rewritten in the form of (10)

$$Q(Y) = \bigvee_{j=1}^{n} (P(u_j) \wedge K(Y, u_j)) \quad (10)$$

Equality (10) is fulfilled only if it is true for any value of the predicate variable Y that spans the set U. Thus, we have the following n equalities (11)

$$Q(u_i) = \bigvee_{n}^{j=1} (P(u_j) \wedge K(u_i, u_j)) \tag{11}$$

for any $i \in 1, \dots, n$.

Defining the values of predicates $Q(u_i)$ and $(P(u_j)$ by y_i and x_j, respectively, where $y_i, x_j \in \{0,1\}$ and $i, j \in 1, \dots, n$. We define the value of the binary predicate $K(u_i, u_j)$ by $k_{ij} \in \{0,1\}$, $i, j \in 1, \dots, n$. Taking into account the following notations, equality (11) will take the form of (12)

$$y_i = \bigvee_{n}^{j=0} (x_j \wedge k_{ij}) \tag{12}$$

It is known that if for arbitrary predicates $P(t)$ and $Q(t)$ exists relation $\psi: P(t) \to X$, than the relation $\psi: (P(t) \vee Q(t)) \to X \vee Y$ is also true. From here we obtain an operator equation of the form (13)

$$K(X) = Y \tag{13}$$

where K is linear logical operator defined on the space $E_{\vee}^{n}$, with the operator matrix (14)

$$K = \begin{vmatrix} k_{11} & \cdots & \cdots & \cdots & k_{1n} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ k_{i1} & \cdots & k_{ij} & \cdots & k_{1n} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ k_{n1} & \cdots & \cdots & \cdots & k_{nn} \end{vmatrix} \tag{14}$$

Thus, the predicate equation (9) is equivalent to the operator equation (13) defined on the logical space $E_{\vee}^{n}$. According to the idea of a continuous matrix type of a linear-logical constant operator acting from space $E_{\vee}^{n}$ in 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 (14) satisfies the above conditions of the idea, then the solution of the equation (13) will be as (15)

$$X = K^{-1}Y \tag{15}$$

The matrix of the inverse operator coincides with the transposed matrix of the operator K. Thus, the solution of the operator equation (15) in the matrix type will be as (16)

$$X = RT * Y \tag{16}$$

As a result of solving the predicate equation (9), it can be written in the form of (17)

$$P(X) = \exists Y (Q(Y) \wedge K(X, Y)) \tag{17}$$

If the operator K is not regular, the solution of the predicate equation cannot be written in the form (17), however, using the algebraic notation of the predicate equation (9), we will look for the solution of the equation in the way defined.

Let's write the operator equation (13) in the form of a system of logical equations (18), assuming that the vector Y is not singular

$$\begin{cases} \bigvee_{n}^{j=1} (k_{1j} \wedge x_j) = y_1 \\ \bigvee_{n}^{j=1} (k_{ij} \wedge x_j) = y_i \\ \bigvee_{n}^{j=1} (k_{nj} \wedge x_j) = y_n \end{cases} \tag{18}$$

Let the ones be worth in Y at places $(d_1, \dots, d_{y^{(1)}}) = D$, and zeros at places $(z_1, \dots, z_{y^{(1)}}) = Z$, $D \cap Z = \emptyset$, $D \cup Z = N$, $N = (1, \dots, n)$. The set of places where the zeros of the vector X are valued will be denoted as $L = (l_1, \dots, l_{X^{(1)}})$. The symbol $*$ will mark the places where there can be zeros or ones. Symbols $*$ stand in places $(m_1, \dots, m_{X^{(*)}}) = M$.

The algorithm is as follows:

1. Init $i = z_1$
2. Form the set consisting of zero coordinates of the vector X
3. Init $j = 1$
4. If $K[i, j] = 1$, then $X_j = l_1$
5. Loop indexes j from 0 to n
6. The index i is set to the next element from the set Z and the transition to clause (9) until all the elements of the set Z are selected

7. Form the set M. Get some logical vector X consisting of zeros and symbols $*$.
8. Check the system (18) for consistency
9. Substitute the found vector into the system
10. Get the solution of the obtained system according to the formula (19)

$$\bigvee_{j=1}^{n} (k_{ij} \wedge x_j) = y_i. \tag{19}$$

11. If the system is not consistent, then the vector is not a solution of the system
12. Substitute 1 instead of the first symbol $*$ In the vector $X(*)$, and zeros instead of the other symbols.
13. Go to step 8
14. If the formed logical vector is a solution of the system, we store it in the array of solutions
15. Set various substitutions of 0 and 1 instead of $*$ symbols, with each new combination going to step 8
16. Write out all the obtained solutions of the system, if the array of solutions is not empty. And if not, then the result is the inconsistency of the system

3.2. Solving logical result problems in databases

The example given 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.

Assume factory f_1 produces parts d_1 and d_2, factory f_2 produces parts d_2 and d_3, factory f_3 produce parts d_1 and d_4, factory f_4 produce parts d_3 and d_4. The existing "factory-part" relationship is easily described by the following binary predicate (20)

$$P_1(f,d) = \begin{cases} 1, \text{if factory } f \text{ produces parts } d \\ 0, \text{in opposite case} \end{cases} \tag{20}$$

where $f \in \{f_1, \dots f_4,\}$ and $d \in \{d_1, \dots d_4,\}$. Thus, information about factories can be stored in the form of a formula record of the predicate $P_1(f,d)$. The next problem is to obtain information about which factory produce part d_1. The corresponding predicate revealing this requirement is written in the second form (21)

$$P_2(d) = \begin{cases} 1, \text{if } d = d_1 \\ 0, \text{in opposite case} \end{cases} \tag{21}$$

As a result, the predicate $P_3(f)$ corresponding to the sought information is denoted by a quantifier equation of the form (22)

$$\exists d\, P_1(f,d) \wedge P_2(d) = P_3(f) \tag{22}$$

and sets the following relation (23)

$$P_3(f) = \begin{cases} 1, \text{if factory } f \text{ produces parts } d_1 \\ 0, \text{in opposite case} \end{cases} \tag{23}$$

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 $E_{\vee}^{n}$. The matrix of operator A has the following form (24)

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{24}$$

and vector $X = (1,0,0,0)$. As a result of the action of the operator A on the vector X, we get the vector $Y = (1,0,1,0)$ and meaning that the part d_1 is served by factories f_1 and f_3. Thus, the operation of searching for information of interest in the database is replaced by the operation of operator multiplication. Now the problem is to calculate which factories produce parts d_1 or d_3. Using the additive property of the linear logical operator A, we have (25)

$$A * X_1 \vee A * X_3 = A(X_1 \vee X_3) = A * X_4 \tag{25}$$

Vectors X_1 and X_3 are created by predicates that formulate the parts d_1 and d_3, respectively. 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, assuming the problem is to calculate which parts are manufactured by factory f_2. Therefore, the logical vector $Y = (0,1,0,0)$. As a result of solving the operator equation of the form of (26)

$$\begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix} * \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} = \begin{vmatrix} 0 \\ 1 \\ 0 \\ 0 \end{vmatrix} \tag{26}$$

relative to X, we get the vectors $(0,1,0,0)$ and $(0,0,1,0)$. So that, factory f_2 produces parts d_2 and d_3.

Next, assume that the database contains information about which parts are used in certain cars. Suppose car c_1 uses part d_2, car c_2 uses parts d_2 and d_3, car c_3 uses parts d_2 and d_3, car c_4 uses parts d_3 and d_4. This relationship "car-part" corresponds to the binary predicate $K_1(c,d)$, defined as (27)

$$K_1(c,d) = \begin{cases} 1, \text{if car } c \text{ uses parts } d \\ 0, \text{in opposite case} \end{cases} \tag{27}$$

where $c \in \{c_1, \dots, c_4\}$ and $d \in \{d_1, \dots, d_4\}$. Similarly, to the previously considered case, it is easy to extract information about which cars 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 (28)

$$\exists d K_1(c,d) \wedge K_2(c) = K_3(d) \tag{28}$$

where the unary predicates $K_2(c)$ and $K_3(d)$ define a specific car and a specific part, respectively. Solving this equation with respect to predicate $K_2(c)$ or predicate $K_3(d)$, we will fetch 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 specific car. The following system of quantifier predicate equations (29) corresponds to this condition

$$\begin{cases} \exists d \ P_1(f,d) \wedge P_2(d) = P_3(f) \\ \exists d \ K_1(c,d) \wedge K_2(c) = P_2(d) \end{cases} \tag{29}$$

Display this system in the form of single equation (30)

$$\exists d \ P_1(f,d) \wedge (\exists d \ K_1(c,d) \wedge K_2(c)) = P_3(f) \tag{30}$$

The quantifier predicate equation (28) corresponds to an operator equation of the form (31)

$$B * T = X \tag{31}$$

in linear logical space E_{V}^n. The logical vectors T and X are constructed, respectively, from the binary predicates $K_2(c)$ and $K_3(d)$. The linear logical operator T has the form matrix (32)

$$\begin{vmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} \tag{32}$$

built on the binary predicate $K_1(c,d)$. Therefore, the predicate equation (30) corresponds to the operator equation of the form (33)

$$\mathrm{A}(\mathrm{B}(\mathrm{T})) = X \tag{33}$$

Transform the resulting equation into the following form (34)

$$F(T) = X \tag{34}$$

where the linear logical operator F is equal to the superposition of operators A and B. In this case, we have (35)

$$C = \begin{vmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{vmatrix} * \begin{vmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{vmatrix} \tag{35}$$

In this way, a rather large search in the database is reduced to the calculation of the matrix of the operator I_F using the operation of multiplying the matrices of the operators A and B.

For example, if we calculate which factories manufacture parts for car c_1. The corresponding logical vector T has the form $(1,0,0,0)$.

As a result, we get that the manufacturers of parts for car c_1 are factories f_1 and f_2.

4. Analysis of possible results

The constant increase in the degree of informatization has created an urgent need for the development of a new theoretical and practical base in the field of formal description of excellent physical information objects. The rapid growth of data volumes in computers, their structural complexity, rapidly progressing computerization, and informatization require a constant increase in the productivity of electronic computing machines, an increase in speed.

Expectations that the role of a universal information mediator will be done by programming languages were not fulfilled, as it became clear that in terms of convenience and flexibility, any artificial language cannot compare with a natural one. At this time, methodological and technical approaches to the creation and use of information systems have already been developed. Currently available intelligent information systems are able to perform functions that were previously considered the exclusive prerogative of humans: prove mathematical theorems, translate texts from one language to another, diagnose diseases and perform many other functions.

Another direction in informatization is the creation of systems of integrated knowledge and the development of methods of active, mental navigation through these systems, including through global computer networks. At this time, the issue of software and hardware methods that effectively manipulate natural language information has turned into such a necessity that the effectiveness of public institutions and production systems begins to depend on. It is no coincidence that among the most popular software tools 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 searching, as well as machine translation programs, optical text recognition, etc. And recently, natural language modules are increasingly being introduced into the operating systems themselves.

All these problems cannot be solved without the involvement of a universal mathematical language. Developments in this field have been underway for several decades, work on the algebraization of logic has been carried out, and a special mathematical apparatus has been developed for the formula representation of relations and operations on them, which is called the algebra of finite predicates. The central place in the algebra of predicates is occupied by relations, they reflect the properties of objects and the connections between them. But until now, there is no convenient method of formulaic declaration of arbitrary relations, which allows them to be implemented programmatically. The possibility of software implementation of formulas that describe predicates or relations is important when designing an automatic control system, when developing a natural-language intelligent interface.

The results of this work, aimed precisely at the creation of modern principled solutions in the construction of methods for the formal presentation of relations using the algebra apparatus of finite predicates, focused on the real calculation of the capabilities of the modern computer and computing base and new requirements for information technologies, can be applied to modeling any logical structures that require a large range of calculations in real time.

The following results were obtained:

- A comparative analysis of methods for solving logical equations was carried out
- Designed and developed a software system that implements the method of solving quantifier linear equations
- The method of solving quantifier linear equations is used to solve the problem of logical results in databases

Although, the work was focused on the modeling of natural language structures, the resulting algorithms have good prospects for application in other fields as well. To describe a given subject area using a system of predicate equations, it is necessary to correctly select a set of semantic features and their values. Semantic signs can be obtained based on the analysis of objects and their properties within the framework of this subject area. By assigning a given object some semantic distinction, we match it with a certain meaning of this attribute.

Based on the work done, it can be concluded that the obtained results can be used in the production of linguistic support for automated information systems, in information search systems, in solving problems of logical results in databases and expert systems, as well as in solving problems of object recognition and classification.

5. Conclusion

Formal methods of intelligent systems were studied and analyzed in the work: methods of displaying knowledge depending on specific fields of application of systems; formal languages that allow representing knowledge in computer memory. Considered areas of application of algebraic methods in automated systems. Found computational practical applications of modern abstract algebra in databases and intelligent systems. Based on the application of algebraic methods in theoretical programming, various translators from high-level languages and various algorithmic algebras were developed.

In order for an arbitrary query to be expressed in any form through basic queries, algebraic operations are introduced on the sets of queries, allowing to operate with queries. Likewise, similar algebraic operations should be introduced on answer sets.

Shown the perspective of using the considered method of solving logical equations in information systems, in particular, in databases. It provides the possibility of obtaining not only information directly stored in the database, but also derivative information obtained on the basis of basic information. The problem of obtaining derivative information is directly related to the problem of the result in intelligent systems, while in the case of applying an algebraic approach to the description of derivative information, a certain algebraic system is distinguished – the algebra of queries, in terms of which derivative information is written through the basic one.

6. References

[1] M. Bur, K. Stirewalt, ORM ontologies with executable derivation rules to support semantic search in large-scale data applications, Proceedings - ACM/IEEE 25th International Conference on Model Driven Engineering Languages and Systems, MODELS 2022: Companion Proceedings, p. 81.

[2] A. Kulik, A. Chukhray, O. Havrylenko, Information Technology for Creating Intelligent Computer Programs for Training in Algorithmic Tasks. Part 1: Mathematical Foundations. System Research and Information Technologies, 2022(4), 27-41. doi:10.20535/SRIT.2308-8893.2021.4.02

[3] Wu Aiyan, Zhang Yongmei, Yang Shang, A Method for Scientific Cultivation Analysis Based on Knowledge Graphs, in: 12th International Conference on Electronics, Communications and Networks, CECNet 2022, p. 13.

[4] Sharonova, N., Kyrychenko, I., Gruzdo, I., Tereshchenko, G., "Generalized Semantic Analysis Algorithm of Natural Language Texts for Various Functional Style Types", 2022 6th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2022), 2022. – CEUR-WS 3171, 2022, ISSN 16130073. - Volume I: Main, PP. 16 - 26.

[5] J. Smith, X. Guo, A. Bansal, A Predicate Construct for Declarative Programming in Imperative Languages, in: PPDP 2022, ACM International Conference Proceeding Series, 2022, article 3551379.

[6] T. Place, M. Zeitoun, A Generic Polynomial Time Approach to Separation by First-Order Logic Without Quantifier Alternation, in: FSTTCS 2022, Leibniz International Proceedings in Informatics, LIPIcs, article 43.

[7] I. Shubin, A. Kozyriev, V. Liashik, G. Chetverykov, Methods of adaptive knowledge testing based on the theory of logical networks, in: Proceedings of the 5th International Conference on Computational Linguistics and Intelligent Systems, COLINS 2021, Lviv, Ukraine, 2021, pp. 1184–1193.

[8] L. Bozzato, T. Eiter, R. Kiesel, Reasoning on Multi-Relational Contextual Hierarchies via Answer Set Programming with Algebraic Measures, in: 37th International Conference on Logic Programming (ICLP 2021), doi: 10.1017/S1471068421000284.

[9] H. Falatiuk, M. Shirokopetleva, Z. Dudar, Investigation of architecture and technology stack for e-archive system, in: 2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 – Proceedings, p. 229.

[10] K. Herud, J Baumeister, Testing Product Configuration Knowledge Bases Declaratively, in: LWDA 2022 - Workshops: Special Interest Group on Knowledge Management (FGWM), Knowledge Discovery, Data Mining, and Machine Learning (FGKD) and Special Interest Group on Database Systems (FGDB), CEUR Workshop Proceedings, vol. 3341, pp. 173-186.