Logical Networks and Their Usage in Solving of Morphological Tasks

Igor Shubin[1[0000-0002-1073-023X]], Andrii Kozyriev[2[0000-0001-6383-5222]], Mariia Pitiukova[3[0000-0002-0978-2622]], Yaroslav Svyatkin[4[0000-0001-6828-6699]]

[1,2,3] Department of Software Engineering
Kharkiv National University of Radioelectronics
Kharkiv, Ukraine
[4] National Technical University "Kharkiv Polytechnic Institute"
igor.shubin@nure.ua, andrii.kozyriev@nure.ua,
mariia.pitiukova@nure.ua, yariks@i.ua

Abstract. In the study of category theory, along with the usual concept of category met. As a result of developing a link between two different definitions of a category and a more general notion of an objectless category, it was found out that a whole class of isomorphic categories with objects corresponds to it.
A universal mathematical apparatus of the algebra of predicates was proposed, and more precisely its central fragment, which refers to the description of logical spaces – logical analysis.
As a result, the interpretation of the category in terms of the algebra of predicates was found – the predicate category $Pred$, and for both cases: the category with objects and the objectless category.

Keywords: Logical Network, Predicate Algebra, Category Theory.

1 Category Theory in Informatics

The problem of automation of software development and determining the structure of the software system by the properties or requirements that it must meet, is only partially solved nowadays. The category theory, the foundations of which were laid by Samuel Eulenberg and Sanders McLain in the early 40's of the twentieth century, deals with the study of object properties relations with its internal structure and, by means of commutative diagrams, expresses the mathematical object structure in its properties. Thus, category theory solves solely internally mathematical problems, and this is a direct output of category theory to the task of software development automation. Note also the fact that commutative diagrams of category theory are very similar to some diagrams of the Universal Modeling Language (UML).

Attempts to apply category theory to informatics began in the 1980s. Category theory was considered as a way of studying system objects. Category theory tools were used in the field of mathematical description of databases. There may be some more examples of the use of category theory in specific fields of informatics, but so far, the

category theory has not yet gained as much ground in informatics as it is in modern mathematics.

Category theory offers a special way of studying objects. There are basically two ways to study the structure of an object. One is to "dissect" its internal content to determine the composition and structure of the parts that make up this object. Another indirect method is to "design" this object to some set of "related" objects and make judgments about the internal structure of the object by properties projections. In fact, the last method, which formalizes within the framework of category theory, seems to be feasible for a complexly organized object.

2 The Mathematical Apparatus of Finite Predicates and Category Theory

Category theory is a special mathematical way of describing objects through their correspondences (morphisms) with each other. It turns out that the properties of a mathematical object (space, group, etc.), which are usually formulated through its internal structure, are quite effectively expressed due to the properties of reflections of this object in the same type of objects. This is an opportunity to translate the study of internal structure into the study of external relations that explains the role of category theory in the study of systemic objects, because the systematic approach offers methods and techniques for the theoretical study of complex objects. Therefore, category theory can be added to the arsenal of systematic approach as a special way of studying objects.

The basis of brain-like hardware and software complexes are logical networks, which in turn are a schematic implementation of predicates algebra formulas which describe algebraic structures. Predicate algebra claims to be a universal mathematical tool for the formal description of information processes. This tool is fully accessible to information system developers for practical applications in improving artificial intelligence. To date, only individual blocks of the lower floors of this algebra have been constructed. However, the requests for informatization urgently require its further intensive development.

In category theory, it is (only in terms of very high levels of abstraction and more generally) essentially the same algebra of predicates. In principle, the difference between category theory and predicate algebra lies only in the fact that the first moves from top to bottom, aimed at knowing the higher logical mechanisms and therefore uses as a starting point the record level of community. The second, moving from the needs of informatization, moves in the study of the same logic of thinking from the bottom up. If it was possible to give a convincing interpretation of the concepts formed by the theory of categories and the methods developed by it in terms of the predicates algebra, that is, specifying, to bring them closer to informatization, then, firstly, it would significantly enrich the toolkit of the predicates algebra, and secondly made it possible to convey a wealth of ideas of category theory to the experts who move forward informatization. And if it was possible to abstract (generalize) the constructs of the predicates algebra, then it would reveal a stimulating effect on the part

of informatization on the development of the theory of categories itself. It is precisely the predicate algebra that we are about to use as such an intermediate domain of knowledge.

It is not easy to use the achievements of category theory in practice for an IT engineer. The abstraction gap is too big. In addition, category theory and informatization are very far apart areas of knowledge. The publications of specialists of mathematicians on category theory do not say a word about linking its content to the needs of informatization. It could be much easier to solve this problem if we were able to find an intermediate area of knowledge of the medium level of abstractness, which links the theory of categories with the practice of informatization, which could serve as a mediator between them.

3 Description of the Network

The logical network is designed to solve the system of equations given by the appropriate model. Analytically, that is, operating with formulas, it is possible to solve any system of equations of predicate algebra that characterizes a given model. It is possible to set any knowledge about the value of any subject variables and to obtain knowledge about the value of any other subject variables. The analytical method works without fail. The main task is to develop such methods for constructing logical networks so that these networks automatically solve any system of equations of the predicate algebra as smoothly as a person can do, using formulas of the algebra of predicates.

The logical network replicates human actions, but the only difference is that people act in a consistent manner, and the network – in parallel. The network works by cycles. Each cycle is divided into two half-cycles – the first and the second. In the first half of the *i*-th cycle, the network for each of its equations of the form of $BEFORE(\mathrm{x},\mathrm{y}) = 1$ ($BEFORE$ – is a relation, set by equation) which finds:

1. By the knowledge of $P_i(\mathrm{x})$ about the value of variable x on the beginning of the *i*-th cycle the knowledge of $Q'_i(\mathrm{y})$ about the value of variable y on the end of *i*-th cycle;
2. By the knowledge of $Q_i(\mathrm{x})$ about the value of variable y on the beginning of the *i*-th cycle the knowledge of $P'_i(\mathrm{y})$ about the value of variable x on the end of *i*-th cycle;

Mathematically, these two operations are expressed by the formulas:

$$\exists x \in A(K(x,y)P_i(x)) = Q'_i(y) \tag{1}$$

$$\exists x \in B(K(x,y)Q_i(y)) = P'_i(x) \tag{2}$$

Here, A and B are the regions of change of the variables x and y. As you can see, every branch of the network is a two-way road.

In the second half of each cycle, the network finds a common part $P_{i+1}(x)$ of all knowledges $P'_{i1}(x), P'_{i2}(x), \dots, P'_{il}(x)$ about the value of each of their subject varia-

bles x sides coming across the nodes of the network from all to the pole x. This operation is expressed as follows:

$$P'_{i1}(x) \wedge P'_{i2}(x) \wedge \ldots \wedge P'_{il}(x) = P_{i+1}(x) \quad (3)$$

The knowledge $P_{i+1}(x)$ then used as the pole state x at the start of the $i + 1st$ cycle. The symbol l indicates the number of branches that approach the pole x. By the beginning of the $i + 1st$ cycle, knowledge of the $P_{i+1}(x)$ deflection is formed in each pole, which is always included in the knowledge of the set $P_i(x)$, which was contained in the same pole at the beginning of the *i*-th cycle. Therefore, the only result of the logical network is to refine the knowledge contained in all its poles according to the source data.

The network is solved by using threads, each thread is tied to a specific pole and intersects all related branches of the pole on the sub-table of the relationship table for the branches. In the process of intersection, many intersections are created on the basis of the table of relations of the branch, and the state is connected with the flow of the pole, after which a logical multiplication is made between the non-digit of the intersection and the set of the state of the intersecting pole, and the result is written in the state of the pole. The process of synchronization of threads takes place, only those threads whose state of poles have changed since the last run of threads are started. If no one of the pole threads is changed, all threads halt.

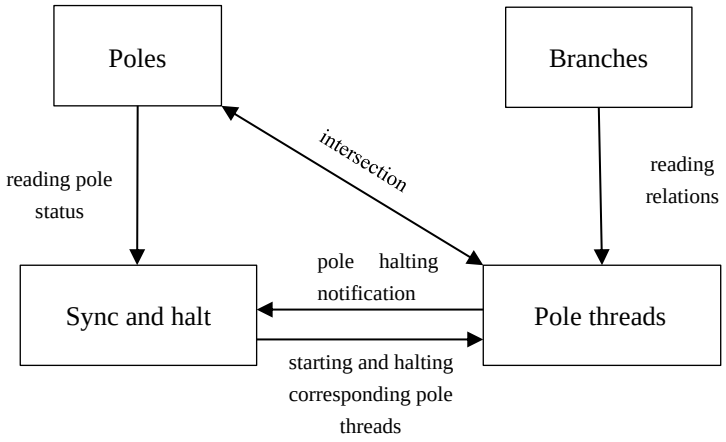

Fig. 1. Conceptual model of multithreading network solution

In order to understand logical analysis, one must disassemble the doctrine of logical spaces, which is the central fragment of the mathematical apparatus of the algebra of predicates.

3.1 Finite Predicates Algebra

The term "predicate algebra" expresses a collective concept. Predicate algebra is not one, but a whole family of algebras. The narrowing of this family begins with the choice of the set U, called the universe of algebra of predicates. In the role of U, you can select any set of elements called objects. Predicate algebras are divided into finite predicate algebra and predicate operations. Algebras of finite predicates are intended for the formulaic notation of fixed predicates, and algebra for predicate operations – for the formulaic expression of operations on predicates.

The transition to a specific variant $\mathcal{A}$ of the algebra of finite predicates is made by fixing the set $(A_1, A_2, \dots, A_m)$ of sets $A_1, A_2, \dots, A_m \subseteq U$. These sets are used as the coordinate axes of the object space $A = A_1 \times A_2 \times \dots \times A_m$ of algebra $\mathcal{A}$. The number m is called the dimension of the object space, and the sets $(a_1, a_2, \dots, a_m) \in A$ of the objects $a_1 \in A_1, a_2 \in A_2, \dots, a_m \in A_m$ are its vectors. We introduce the subject variables $x_1 \in A_1, x_2 \in A_2, \dots, x_m \in A_m$ and the logical variable $t \in \Sigma$, where $\Sigma = \{0,1\}$. The elements of the set are called logical when 0 is false and 1 is true.

Predicate P on the set $A = A_1 \times A_2 \times \dots \times A_m$ is called as any function $P(x_1, x_1, \dots, x_m) = t$, which matches each vector $(x_1, x_1, \dots, x_m)$ of the space A some value t from set Σ. Thus, predicate P displays the set A to the set Σ, that is $P: A_1 \times A_2 \times \dots \times A_m \to \Sigma$. The set $\mathcal{M}$ of all $P: A \to \Sigma$ used as a carrier of finite predicate algebra $\mathcal{A}$.

On the set Σ negation operations $\overline{\xi}$ are defined of the logical element ξ $\overline{0} = 1, \overline{1} = 0$, and also the disjunction $\xi \vee \eta$ and conjunction $\xi \wedge \eta = \xi$ of the logical elements ξ and η

- $0 \vee 0 = 0$;
- $0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$;
- $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0$;
- $1 \wedge 1 = 1$.

The set $\mathcal{M}$ denotes operations of negation $\overline{P}$ of the predicate P

$$\overline{P}(x_1, x_2, \dots, x_m) = \overline{P(x_1, x_2, \dots, x_m)} \tag{4}$$

and also, disjunction $P \vee Q$ and conjunction $P \wedge Q = P$ of predicates P and Q

$$(P \vee Q)(x_1, x_2, \dots, x_m) = P(x_1, x_2, \dots, x_m) \vee Q(x_1, x_2, \dots, x_m), \tag{5}$$

$$(P \wedge Q)(x_1, x_2, \dots, x_m) = P(x_1, x_2, \dots, x_m) \wedge Q(x_1, x_2, \dots, x_m). \tag{6}$$

These operations play a fundamental role in the algebra of finite predicates $\mathcal{A}$. Operations of disjunction, conjunction and negation of predicates obey the laws of:

— idempotency $P \vee P = P, PP = P$;
— commutability $P \vee Q = Q \vee P$, $PQ = QP$;
— associativity $(P \vee Q) \vee R = P \vee (Q \vee R)$, $(PQ)R = P(QR)$;
— distributivity $(P \vee Q)R = PR \vee QR$, $PQ \vee R = (P \vee R)(Q \vee R)$;
— exclusion $P \vee PQ = P$, $P(P \vee Q) = P$;

– collapsing $P \vee Q\overline{Q} = P$, $P(Q \vee \overline{Q}) = P$;
– double negation $\overline{\overline{P}} = P$;
– de Morgan's law $\overline{P \vee Q} = \overline{PQ}$, $(P, Q, R \in \mathcal{M})$.

In the role of basic elements in finite predicate algebra, the subject's object predicates are used

$$x_i^a = \begin{cases} 1, & \text{if } x_i = a \\ 0, & \text{if } x_i \neq a \end{cases} \tag{7}$$

$i = \overline{1,m}$, $a_i \in A_i$, as well as predicate 1 equals one, and predicate 0 equals zero.

The predicates of knowing the subject obey the law of truth

$$\bigvee_{a \in A_i} x_i^a = 1, (i = \overline{1,m}), \tag{8}$$

the law of error, where $a \neq b$

$$x_i a \, x_i b = 0, \ (i = \overline{1,m}, a, b \in A_i), \tag{9}$$

and the law of negation

$$\overline{x_i^a} = \bigvee_{\substack{b \in A_i \\ b \neq a}} x_i^b, (i = \overline{1,m}; a \in A_i) \tag{10}$$

The predicates 0 and 1 obey the laws of false $0 \vee A = A$, truth $1 \vee A = 1$, contradiction $A\overline{A} = 0$ and the exclusion of the third $A \vee \overline{A} = 1$ $(A \in \mathcal{M})$.

Theorem on completeness of the basis of the algebra of finite predicates. *Any presavage P on A is expressed in the algebra of finite predicates $\mathcal{A}$ in the form*

$$P(x_1, x_2, \dots, x_m) = \bigvee_{\substack{a_1 \in A_1 \\ a_2 \in A_2 \\ \dots \\ a_m \in A_m}} (P(a_1, a_2, \dots, a_m) x_1^{a_1} x_2^{a_2} \dots x_m^{a_m}) \tag{11}$$

The formula to the right of the equality sign is called the disjunctive normal form of the predicate P.

3.2 Objectless Categories

Next, let's consider the objectless categories. Let K be any objectless category. The morphisms of a nonobjective category K are understood to mean elements of some arbitrarily chosen $MorK$. Morphisms, as before, are denoted by lowercase letters from the middle of the Latin alphabet. Instead, $f \in MorK$ is sometimes written $f \in K$ for

brevity. In the $MorK$ set, a binary partial operation of multiplying fg by the morphisms $f,\ g \in K$ is defined. The product of morphisms is associative: $(fg)\,h = f(gh)$ for any morphisms f, g, h of category K whenever there are $(fg)h$ and $f(gh)$ morphisms.

Each morphism $e \in MorK$ is called identical in category K if $ee = e$. If for any morphisms $f, g \in MorK$ for which the products fe and eg exist, then equals $fe = f$ and $eg = g$ are true. The identity morphism e is called right for morphism f if $fe = f$, and left for morphism g if $eg = g$. For each morphism f of category K, there are single right and only left identical morphisms. The product fg of the morphisms f, g of category K exists if and only if the right identical morphism of morphism f coincides with the left identical morphism of morphism g.

An objectless category K is called a set of $MorK$, in which a partial operation $MorK \times MorK \to MorK\ fg$ is given, which satisfies the conditions:

1. For any $f, g, h \in MorK\ \ (fg)h = f(gh)$ if and only when $(fg)h, f(gh) \in MorK$;
2. For any $f \in Mork$ there are single right and only left identical morphisms;
3. The product fg of morphisms f, g of category K exists if and only if the right identical morphism of morphism f coincides with the left identical morphism of morphism g.

Let's consider the nature of defining an objectless category to a category with objects. Denote by IdK the set of all identical morphisms of an objectless category K. Let's work in the set of ObK objects of category K, an equilibrium set of IdK. We assume that the set IdK is mapped to the set ObK by the bijection Φ. We assign to each $f \in MorK$ morphism a single pair of objects $(\Phi(e), \Phi(e'))$, where e is left and e' is the right identical morphisms of morphism f. The object $A = \Phi(e)$ will be called the beginning of the morphism f, and the object $B = \Phi(e)$ its end and write $f\colon A \to B$. It can be proved that in such a refinement of the objectless category, we come to the definition of the category of objects described above. A single objectless category K corresponds to a whole family of categories with objects $\{K_\Phi\}$, where $\Phi\colon IdK \to ObK$ is any bijection. There is a single restriction on the choice of the ObK set: $|idk| = |obk|$. All categories of $\{K_\Phi\}$ are isomorphic to each other.

3.3 Logical Field

For the convenience of the following interpretation of the category concept described above in terms of the predicate algebra, we will need to present the finite predicate algebra in the abstract as a logical space. Let G be a nonempty set called a logical field (in short – a field). The elements of the set G are called logical scalars (in short – scalars). Scalars will be denoted by Greek letters.

The set $G \times G$ defines an operation $\alpha \vee \beta$ with values in the set G called the disjunction or logical addition (in short – addition) of the scalars α and β. To add scalars, the following axioms are fulfilled:

— the law of idempotency – for each $\alpha \in G\ \alpha \vee a = \alpha$;
— the law of commutativity – for any $\alpha, \beta \in G\ \alpha \vee \beta = \beta \vee \alpha$;

— the law of associativity – for any $\alpha, \beta, \gamma \in G\ \alpha \vee (\beta \vee \gamma) = (\alpha \vee \beta) \vee \gamma$;
— zero law – there is a unique element $0 \in G$ such, that for each $\alpha \in G\ \ 0 \vee \alpha = \alpha$
— the law of unity – there is a unique element $1 \in G$ such that for each $\alpha \in G\ \ 1 \vee \alpha = \alpha$.

A scalar 0 is called a zero scalar or a zero of a field G, scalar 1 is a single scalar or one of a field G.

The set $G \times G$ defines the operation $\alpha \wedge \beta = \alpha\beta$ with values in the set G called conjunction or logical multiplication (in short – multiplication) of the scalars α and β. The following axioms are fulfilled for the multiplication of scalars:

— the law of idempotency – for each $\alpha \in G\ \alpha a = \alpha$;
— the law of commutativity – for any $\alpha, \beta \in G\ \alpha\beta = \beta\alpha$;
— the law of associativity – for any $\alpha, \beta, \gamma \in G\ \alpha(\beta\gamma) = (\alpha\beta)\gamma)$;;
— zero law – for each $\alpha \in G\ \ 0\alpha = 0$;
— the law of unity – for each $\alpha \in G\ \ 1\alpha = \alpha$.

The following axioms link the addition and multiplication of scalars:

— the law of distributivity – for any $\alpha, \beta, \gamma \in G\ \ \alpha(\beta \vee \gamma) = \alpha\beta \vee \alpha\gamma, \alpha \vee \beta\gamma = (\alpha \vee \beta)(\alpha \vee \gamma)$;
— the law of elimination – for any $\alpha, \beta \in G\ \ \alpha \vee \alpha\beta = \alpha, \alpha(\alpha \vee \beta) = \alpha$

The set G defines a single operation $\overline{\alpha}$ with values in the set G called the negation of the scalar α. To deny the scalar, the following axioms are satisfied:

— the law of double negation – for each $\alpha \in G\ \overline{\overline{\alpha}} = \alpha$;
— the law of negation of zero – $\overline{0} = 1$;
— the law of negation of the unit – $\overline{1} = 0$.

The following axioms link objections to the addition and multiplication of scalars:

— the law of exclusion of the third – for each $\alpha \in G\ \ \alpha \vee \overline{\alpha} = 1$;
— the law of contradiction – for each $\alpha \in G\ \ \alpha\overline{\alpha} = 0$;
— de Morgan's laws – for any $\alpha, \beta \in G\ \ \overline{\alpha \vee \beta} = \overline{\alpha\beta}$;
— the law of minimization – for any $\alpha, \beta \in G\ \ \alpha \vee \beta\overline{\beta} = \alpha,\ \ \alpha(\beta \vee \overline{\beta}) = \alpha$

The above laws are called axioms of the logical field. Taken by itself, the logical field can be identified with the Boolean algebra.

3.4 Logical Space

Let M be a nonempty set called a logical vector space over a field G (short is a vector space, a logical space, or a simple space). The elements of the set M are called logical vectors (in short – vectors). We will denote the vectors with lowercase Latin letters. The set $M \times M$ defines the operation $a \vee b$ with values in the set M, called disjunction

or logical addition of vectors a and b. To add vectors, the following axioms are satisfied:

- the law of idempotency – for any $a \in M\ a \vee a = a$;
- the law of commutativity – for any $a, b \in M\ a \vee b = b \vee a$;
- the law of associativity – for any $a, b, c \in M\ a \vee (b \vee c) = (a \vee b) \vee c$;
- zero law – there exists a unique element $0 \in M$ such that for any $a \in M\ 0 \vee a = a$;
- law of unity – there is a unique element $1 \in M$ such that for any $a \in M\ 1 \vee a = 1$.

The vector 0 is called the zero vector or zero of space M, vector 1 is called the single vector or unit of space M.

On the set $G \times M$, the operation $\alpha \wedge a = \alpha a$ with values in the set M, called conjunction or logical multiplication of the scalar α by vector a is defined. The operations of scalar multiplication and scalar multiplication by vector associate with each other the following axiom:

- the law of associativity – for any $\alpha, \beta \in G$, and any $a \in M$ $(\alpha\beta)a = \alpha(\beta a)$.

The operations of adding scalars and vectors and multiplying a scalar by a vector are related by axioms:

- the law of left distributivity – for any $\alpha, \beta \in G$ and any $a \in M$ $(\alpha \vee \beta)a = \alpha a \vee \beta a$;
- the law of right distribution – for every $\alpha \in G$ and any $a, b \in M$ $\alpha(a \vee b) = \alpha a \vee \alpha b$;
- zero law for any $a \in M$ $0a = 0$;
- the law of unity is for any $a \in M$ $1a = a$.

The axioms of a logical field, together with the laws just given, are called axioms of a logical space. The operations of adding scalars and vectors are denoted by the same sign. Such homonymy of the sign of addition does not lead to confusion, however, since its content is easily specified in context. The same applies to the operation of multiplication of scalars and multiplication of scalar by vector, as well as zero (singular) is a scalar and a vector. Logical space is otherwise called logical algebra. Logical algebra has some similarity to linear algebra. The notion of logical space corresponds to the notion of linear space in linear algebra. linear space is sometimes called linear analysis, and by analogy with this, the doctrine of the properties of logical space is called logical analysis. The term "logical analysis" is also used by Russell to refer to the science of logical means as a study of logical means. Intelligence: It seems to us that logical algebra can serve as a mathematical tool through which such a study can be successfully conducted.

The combination of vectors $a_1, a_1, \ldots, a_m$ (not necessarily different) is called a vector u equal to

$$u = \alpha_1 a_1 \vee \alpha_2 a_2 \vee \ldots \vee \alpha_m a_m = \bigvee_{k=1}^{m} \alpha_k a_k \qquad (12)$$

Here, $\alpha_1, \alpha_2, \ldots, \alpha_m$ are any scalars called combination coefficients. The combination of vectors $a_1, a_1, \ldots, a_m$ will again be a combination of the same vectors. A combination of vectors is called trivial if all its coefficients are zero and non-trivial – if not, then. The trivial combination of any vectors is zero. If the vector u is a combination of vectors $a_1, a_1, \ldots, a_m$, then we will say that it depends on them. The zero vector depends on any nonempty set of vectors. If the vector u cannot be represented as any combination of vectors $a_1, a_1, \ldots, a_m$, then we will say that it is independent of them.

We say that the empty system of vectors $\{ak\}^{mk} = 1 = \{a_1, a_2, \ldots, a_m\}$ is independent if each of the vectors that are included in the system does not depend on its other vectors. Any system consisting of a single non-zero vector is considered independent. Vector 0 is not part of any independent vector system that contains non-zero vectors. If at least one of the vectors in the system depends on its other vectors, then we will say that such a system of vectors is dependent. If the set of vectors $\{a_1, a_2, \ldots, a_m\}$ is independent, and the set of vectors $\{a_1, a_2, \ldots, a_m, a_{m+1}\}$ is dependent, then the vector a_{m+1} is a combination of vectors $\alpha_1, \alpha_2, \ldots, \alpha_m$.

A set of vectors is called generating if all the vectors of space M are their combinations. It is easy to prove that any minimal (by the number of vectors) generating set of vectors is independent.

A logical space M is said to be finite-dimensional if it contains a finite number r of vectors $e_1, e_2, \ldots, e_r$, through which it can be expressed as

$$x = \bigvee_{i=1}^{r} \alpha_i e_i \tag{13}$$

any vector $x \in M$, where $\alpha_1, \alpha_2, \ldots, \alpha_m$ are some coefficients. In other words, the logical space M is finite-dimensional if there exists at least one finite generating system $\{e_1, e_2, \ldots, e_r\}$ for it. And if not, the logical space is called infinite-dimensional. The least of the numbers r satisfying condition (2), (4) is called the dimension or number of dimensions of the logical space. If the dimension of the logical space is n, then we say that the space M is n-dimensional. If the space M is n-dimensional, then it contains n independent vectors. In a dimensional logical space, any generating set of vectors containing n elements is called a basis. Any finite logical space is finite-dimensional and has a finite basis in it.

3.5 Network Solution Algorithm

There are many binary relationships. This set is a system because it is a decomposition of one relation. This whole system can be represented as a non-oriented graph and is called a logical network. The edges of this graph are called branches of the network, the vertices are called poles. Branches of the logical network represent the already mentioned binary relations. Network poles are memory cells, each of which holds many values of some object variable. The set of all values of a variable is a domain, any subset of it will be a unary relation; this subset characterizes the knowledge of the value of a given subject variable contained in a given pole. The subject variable pole can still be called an attribute. Initial conditions in some poles indicate subsets of domains or knowledge of a variable, that is, set of many attribute

values. It turns out that we narrow down the set of values for each pole where the information is stored. It is necessary to determine how this will affect the values of those variables stored in other poles. In other words: "It is possible to set any knowledge about the value of any subject variables and to obtain knowledge about the value of any other subject variables."

Each binary relation (network branch) is described by a predicate equation of the form $P_i(x, y) = 1$. For programmatic processing of relations, it is convenient to represent tables with two columns (these are relational relations with two attributes). Since we have to calculate the values of the variables in the equation, it turns out that we solve this equation. Many variables in the pole of the network are described by the predicate equation $C_j(x) = 1$. This set can be represented by a single-column table that lists the values of a given variable (a relational relationship that has one attribute). Substituting knowledge into a logical network and computing knowledge for variables of interest is a simple solution to the system of predicate equations, each of which contains two variables. In relational terms, the task can be described as follows.

There is a system of binary relationships that have common attributes. As a condition of the task are restrictions on some of the attributes that are set by unary relations (i.e., with one attribute). These restrictions are set by the user. The result of the network, that is, the source data, will be many unary relations – many values for each of the attributes.

Strictly speaking, the graph of the logical network is oriented, since each undirected edge implies a pair of oriented, directed in opposite directions.

The main area of application of this software product is the development of logical networks and their testing, for later use in systems related to artificial intelligence, and the creation of high-speed maps, as well as individual application modules can be used by software developers to create artificial intelligence systems based on binary logical networks.

The system can be modified in the direction of exception constraints, and modules can be added to export the network to known DBMS.

4 Logical Network Operation in Morphological Task

The task is to develop methods building logical networks so that they automatically solve any system of equations of predicate algebra can be done by a person using the formulas of predicate algebra.

The logical network copies the actions of a person, but the only difference being that the network works in – in parallel on a clock basis. Each measure is divided into two half-cycles – the first and second. In the first half-cycle of the i-th cycle, the network for each of its equations of the form of $K(x, y) = 1$.

In the first half-cycle it searches for 1) a known knowledge $P_i(x)$ about the value of the variable x in at the beginning of the i-th step, the knowledge of $Q'_i(y)$ about the value of the variable y at the end of the i-th beat; 2) according to the known knowledge of $Q_i(y)$ about the value of the variable y at the beginning of the i-th measure; knowledge of $P'_i(x)$ about the value variable x at the end of the i-th step.

In the second half-cycle of each measure, the network searches for the common part $P_i + 1(x)$ of all knowledge $P'_{i1}(x), P'_{i1}(x), \ldots, P'_{il}(x)$ about the value of each of its subject variables x, coming along the network branches from all sides to the pole x.

In the role of the state of the pole, knowledge $P_{i+1}(x)$ is used at the initial moment of the $i + 1$-st cycle. The number of branches approaching the pole x is denoted by the symbol 1. In each pole, at the beginning of the $i + 1$-st beat, each generates $P_{i+1}(x)$, which belongs to the set $P_i(x)$, which was contained in the pole at the beginning of the i-th step. We can conclude that the result of this logical network is the refinement of knowledge that is contained in all poles of this network in accordance with the source data.

A lot of binary relations is a system, since it is a decomposition of one relationship. This system is presented in the form of an undirected graph, called a logical network. The edges of this graph are network branches, the vertices are poles. The branches of this logical network are the binary relations that is mentioned above. Network poles are cells memory, each of which contains many values of some subject variable. a Domain is a set of values of a variable, and any subset of it is a unary relation. An attribute is an object variable of a pole. The set of attribute values or a subset of domains is indicated as initial conditions at some poles. Thus, the set of values for each pole narrows, and at the same time, you can set any knowledge about the values of any subject variables and get knowledge of the values of any other subject variables.

In relational terms, the problem of solving a system of predicate equations can be described as follows:

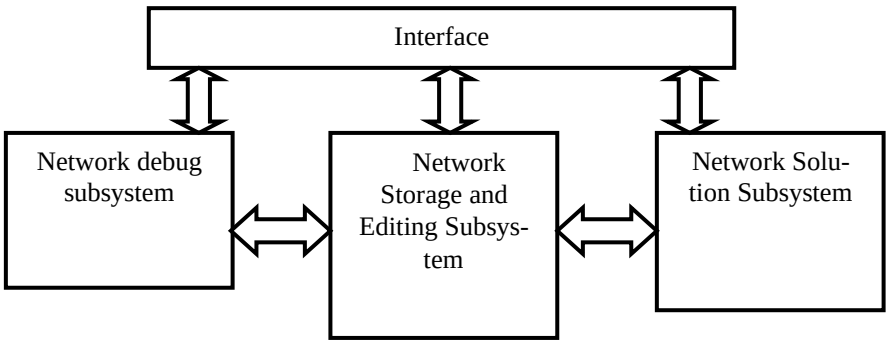

Fig. 2. Diagram of a software model for solving a network

First, we set the initial values of some poles.

1. We determine the branches along which the movement of information will take place. These branches determine the names of the equations that need to be solved in this measure. In order for the information to go to all adjacent poles, it is necessary to determine the branches incident to the active poles on a common network graph or table of branches.

2. In order to solve the equation, we need to convert the set of values on the first half-cycle of this measure. To do this, substitute all contained in the active pole x x_cur data into the equation expressed by the ratio P_i.
3. We define the set of all poles adjacent to the active.
4. In the second half-cycle, we determine the value of each adjacent pole y, i.e. fill all adjacent poles.
5. Check if the stability criterion is reached: for each new value of the pole y_cur obtained on this measure, there is an old value for this pole y_old, and these values match.
6. All the poles that were filled on this measure become active for next measure. All other poles cease to be active.
7. Go to the next measure (point 1)

The network functions as a kind of database. In the role of database attributes in networks are subject variables represented by network poles. The role of the domain of database attributes in the network are the areas of change of subject variables network. The tables stored in the database are binary relations represented by network branches. In the role of the source data entering the database for the solved problem is a subset of some of the areas of change in the subject variables of the network. As a query to the database are some of the subject network variables. The user can specify any set of interests subject variables. These object variables correspond to those poles with which information is taken after the network has decided her task. The information provided by the network has the form of subsets of areas of change subject variables specified by the network user.

5 Conclusion

The algebra of finite predicates is represented abstractly in the form logical space; logical scalars, vectors and operations on them, as well as axioms of a logical field are introduced in the logical space.

Developed in terms of finite predicate algebra Interpretations of some concepts of category theory made it possible to use the apparatus of category theory together with the apparatus of algebra of finite predicates to create information systems, in particular, when constructing logical networks.

The developed logical network allows us to reduce the area of change for intermediate variables, and also extends the coverage of morphological problems modeled by logical networks.

References

1. Bondarenko M., Dudar Z., Protsay V., Cherkashin V., Chikina V., Shabanov-Kushnarenko Yu.: Algebra of predicates and predicate operations. In: Radio electronics and computer science. №1, pp. 5-22. (2004).

2. Shabanov-Kushnarenko Yu., Klimushev V., Lukashenko O.: Brainlike computing. In: Proceedings of the East-West Design and Test Workshop. Odessa, Ukraine, pp. 274-279. (2005).
3. Dudar Z., Ivanilov V., Klimushev V., Obrizan V.: Logical network for the verb inflection model of the Russian language. In: Eastern European Journal of Advanced Technologies. Kharkiv, №4/2, pp. 80-89. (2006).
4. Shubin Yu., Pitiukova M.: Logical measures and usage for the most morphological tasks. In: Proceedings of the 3rd International Conference “Innovative Technologies in Science and Education, pp. 402-405. Amsterdam, The Netherlands (2019).
5. Gabbay M., Mulligan D.: Nominal Henkin Semantics: simply-typed lambda-calculus models in nominal sets. In: LFMTP 2011, vol. 71 of EPTCS, Proceedings of the 6th International Workshop on Logical Frameworks and Meta-Languages, pp. 58-75. (2011).
6. Peters S., Zhang C., Livny M., R´e C.: A Machine Reading System for Assembling Synthetic Paleontological Databases. In: PLoS One (2014).
7. Bondarenko M., Leshchynska I., Kruglikova N., Rusakova N., Shabanov-Kushnarenko Yu.: About relational networks. In: Bionics of Intelligense: Sci. Mag. (2010).
8. Sabek I., Musleh M., Mokbel M.: Flash in Action: Scalable Spatial Data Analysis Using Markov Logic Networks. In: VLDB (2019).