ISSN 0147-6882, Scientific and Technical Information Processing, 2025, Vol. 52, No. 6, pp. 587–597. © Allerton Press, Inc., 2025.
Russian Text © The Author(s), 2024, published in Iskusstvennyi Intellekt i Prinyatie Reshenii, 2024, No. 3, pp. 18–31.

Notional Model of Knowledge

V. S. Vykhovanets[a, *]

[a] *Bauman Moscow State Technical University, Moscow, Russia*

**e-mail: valery@vykhovanets.ru*

Received March 24, 2024

Abstract—This article discusses well-known methods of knowledge representation and processing. A new model, called notional, is proposed, characterized in that the relations (links) between notions are considered as ordinary notions. The notion is considered as a form of thought expressed by a named set of entities. A specific notion identifies one entity. An abstract notion is formed from other notions by generalization (union) and association (Cartesian product) them. Declarative knowledge is given by enumerable sets of entities, and procedural knowledge is given by solvable ones, where resolving procedures are expressed by formulas of the pure monadic predicate calculus. The description of the applied language of knowledge representation and processing is given. It is proven that queries to the notional model are executed in polynomial time from the logarithm of the average number of entities in notions. The possibility of unsupervised learning of notional models is substantiated. It is shown that the notional model makes it possible to visually represent and effectively process both declarative and procedural knowledge.

Keywords: knowledge representation, knowledge processing, knowledge models, notional model, declarative knowledge, procedural knowledge, unsupervised learning, intelligent information system

DOI: 10.3103/S0147688225700467

INTRODUCTION

Knowledge processing, as a task of artificial intelligence, consists in, using technical means, simulating the acquisition, accumulation, organization, adaptation, actualization, and use of knowledge by humans [1].

The existing problems in the field of knowledge representation and processing are expressed in the fact that none of the known knowledge models allows for the implementation and efficient processing of large volumes of knowledge. The reason for these problems is that the basic part of all knowledge models is an oriented graph, the vertices of which are notions, and the arcs define the relationships between them. Hence, any request to the knowledge base is a search in the oriented graph for a subgraph with given properties (an isomorphic graph), which in the general case is an NP-complete problem [2].

Recently, there has been a tendency to use artificial neural networks to solve artificial intelligence problems. However, artificial neural networks are also unsuitable for knowledge representation and processing, since it is impossible to explain the results obtained when using such networks, there is no guarantee of obtaining the correct (complete) solution to the problem, the predictive ability of the network is unpredictable, there are difficulties in solving problems step by step and computational problems requiring long training of the network and high costs for obtaining a training sample, there is a low quality of training with incomplete or noisy training data, there is a risk of overfitting the network and dependence of the network structure on the problem being solved, there is inadequate behavior of the network in dynamically changing subject areas, etc.

The goal of this article is to analyze the known models of knowledge representation and processing and determine their advantages and disadvantages. It is proposed to use a notional model of knowledge that is free from many shortcomings and is distinguished by its not providing a division into notions and relationships between them. For this reason, the mathematical language of the notional model is the pure monadic predicate calculus, which expresses both declarative and procedural knowledge. The possibility and efficiency of using the notional model for representing and processing large volumes of knowledge are substantiated.

1. STRUCTURE OF KNOWLEDGE

In the most general sense, knowledge is a mental entity aimed at adequately reflecting the surrounding reality in the human consciousness, which is subjective (ideal) by its nature and cannot be directly studied [3]. Knowledge is formed as a result of the pedagogical process, self-education, and life experience.

Knowledge is manifested as a result of cognitive activity and is logically or factually substantiated and allows verification for reliability. The presentation of knowledge is the alienation of knowledge from its

bearer in some external (material) form, and the objectification of knowledge is the recognition of knowledge presented in an external form as objectively reliable and socially significant.

Knowledge as a result of abstract logical thinking is recorded in human consciousness in the form of notions, judgments, inferences, reasoning, theories, etc.

A notion is a form of thinking that singles out and names something.

A judgment is a form of thinking that reveals the connection between notions. Complex judgments are obtained as a result of inferences and can consist of two or more simple ones.

Inference is a form of thinking that links premise judgments with consequence judgments. Depending on cultural tradition, level of education, mental state, etc., the rules for making inferences may differ. Inferences must transfer the reliability (truth, plausibility, probability, analogy, persuasiveness, argumentation, etc.) of the initial judgments to the resulting one [4]. The rules of deductive inferences formulated in logic are considered objective. Another example is inductive and abductive inferences, which transfer plausibility, but not truth, to consequence judgments [5].

Reasoning is a mental activity aimed at obtaining consequence judgments from premise judgments. Depending on the rules used to perform inferences, deductive, inductive, and abductive reasoning, reasoning based on precedents, reasoning by analogy, probabilistic and nonmonotonic reasoning, and other types of reasoning are used.

In turn, theory is consequence judgments obtained in the process of reasoning from some premise judgments with the help of inferences of a given type.

Thus, knowledge is the result of a rational reflection of reality in the consciousness of a person, which is presented in an external form by notions and judgments approved by a certain set of inferences.

2. KNOWLEDGE MODELS

A knowledge model is a means of formally describing knowledge (knowledge language) and methods of operating with it (inference on knowledge). There are several types of knowledge model:

– logical models (language of logic): predicate calculus [6], description logics [7];

– production models (language of productions) [8];

– network models (language of directed graphs): semantic networks [2], conceptual graphs [9];

– frame models (object language): object (frame) models [10], relational models [11];

– notional models (language of notions) [12].

Logical knowledge models are considered fundamental. However, due to the monotony of their output, it is impossible to revise intermediate results, i.e., everything is facts and there are no hypotheses. It is also almost impossible to use predicates as predicate variables, which limits the expressiveness of a logical language. The undecidability of first-order logic formulas does not allow for reliable recognition of judgments about the subject area. The language of logic does not provide means for expressing procedural knowledge. Logical models are trained only with a teacher. There is a high computational complexity of inference in first-order logic, as well as some limitations and unnaturalness of the logical language.

Production models based on rules of the type "if Condition, then Action" have difficulties in checking for consistency and completeness, there is unpredictability in the inference results due to the nondeterminism of the choice of productions, it is necessary to use heuristics when inferring, there are no means of representing procedural knowledge, and models are trained only with a teacher.

The frame model is quite visual, but it does not have an inference control mechanism, there is a high structural complexity of data, poor modifiability of knowledge, and a low inference speed requiring the use of heuristics is observed. Frame models are trained only with a teacher.

Semantic networks are the most common for knowledge representation. A semantic network consists of a large number of notions (nodes) and relationships between them (arcs). A semantic network does not directly contain the structure of the subject area, which leads to the complexity of its modification and updating. The inference procedure on the network cannot guarantee the reliability of the result, and the result itself has no logical justification. As a result, inference in a network model can be only approximate, implemented using heuristics. Just as with logical models, semantic networks are trained only with a teacher and do not have the means to represent procedural knowledge.

Many of the listed shortcomings of knowledge models are absent in notional models.

3. NOTION AND CONCEPT

The main assumption used in notional models is the difference between a notion and a concept. A notion is a simple idea, opinion, representation, or understanding of something; notional is hypothetical, imaginary [13, p. 538]. A notion is related to representations from a subject area—entities. A notion differs from a concept.

A concept is a general notion, an abstract idea formed by combining all aspects of a certain class of objects [13, p. 159]. A concept consists of instances, each of which is characterized by a set of properties, and properties can be defined and nondefined [14, p. 38]. Thus, a concept is objectified by its definition, and a notion is subjective.

The problems of unambiguous and consistent description of concepts associated with the variability of the set of properties of instances have led to the emergence of alternative theories of conceptual modeling [14, p. 39]. Some of these theories are based on a probabilistic approach, in which the set of properties is expanded and their appearance in instances of the concept has a probabilistic nature. Other theories divide the properties of instances of the concept into typical ones, inherent in all instances, and atypical ones, which can appear only in some of them. In the extreme case, it is argued that it is not possible to achieve a common understanding of concepts for various social, political, and other reasons [15, p. 334].

In the notional model, the problems that arise in describing concepts are solved on the basis of a corpus approach—by identifying several problem areas in one subject area and forming a corpus of theories [16]. Because the notion is subjective, it is necessary to provide for the possibility of several alternative descriptions of the same notion, in each problem area in its own way. This will allow the notion to be subsequently objectified in its collective (conceptual) form.

A problem area is a subject area considered in a certain narrow sense (aspect), from the point of view of a certain active problem, which structures the analyzed subject area differently each time. The same notion in different aspects (in different problem areas) may have different definitions and descriptions. To name notions in different aspects, the aspect name is added to the notion name, separated by the @ symbol. If we are talking about a concept or a notion in a general aspect, the aspect name with the @ symbol is not added to the notion name.

For example, the Project concept is an objectification of the Project notions in the Object, Plan, Program, Activity, Document aspects, etc. The entities of the Project notion in the Object aspect (the Project@Object notion) are the results of purposeful practical activity; in the Plan aspect (the Project@Plan notion), the entities are developed plans (schemes) of structures, mechanisms, devices, buildings, etc.; in the Program aspect (the Project@Program notion), the entities are a sequence of interrelated works (tasks) to achieve a certain goal within a given time; in the Activity aspect (the Project@Activity notion), the entities are a temporary undertaking aimed at creating a unique product, service, or result; in the Document aspect (the Project@Document notion), the entities are the preliminary text of a document, resolution, etc.

4. NOTION FORMATION

In notional modeling, notions are formalized as named sets of entities of the subject area and are divided into specific and abstract. Specific notions are formed on the basis of representations, and abstract notions are formed by performing operations on notions.

Specific notions or sign notions are formed by identifying (naming) unique representations (entities) in a particular problem area. In the process of interpreting a sign notion, a reverse transition to the entities that it designates is carried out.

Examples of sign notions can be, for example, entities identified by words such as Red, Many, Sometimes, Friendship, etc. The reverse transition from sign notions to entities is carried out when interpreting these words.

Abstract notions are generalizations or associations of other notions. An association notion is formed by Joining (concatenating) the entities of the associated notions into the entity of the association notion, and a generalization notion is formed by combining the entities of the generalized notions.

Examples of generalization notions include the Tree notion, which is a generalization of notions of different species of trees, or the notion of Number, which is a generalization of notions of different types of numbers: Integer, Rational, Real, and Complex.

The inverse operation for generalization is the operation of specialization, in which the notions generalized in the generalization notion are extracted from it.

Typification is a special case of generalization, in which sign notions are subject to unification. The inverse operation for typification is the operation of concretization, in which the notions typified in the type notion are extracted from it. For example, the Time notion is a typification of sign notions denoting different moments in time, and the Natural number notion is a typification of sign notions of natural numbers.

In turn, the Pipe notion is an example of an association notion, combining notions such as Material, Diameter, and Length; however, not all combinations of the material of the pipe, its diameter, and length are acceptable. Another example is the Steel association notion, which connects notions such as Carbon, Iron, Chromium, Nickel, Manganese, and other components in specified mass fractions for each steel grade. As can be seen from the examples, the association connects notions, not entities, i.e., it is not a part–whole relationship between instances, as is the case in other models.

The inverse operation for association is individualization, in which the notions associated in the association notion are extracted from it. A special case of association is the aggregation of notions. During aggregation, the entities of the aggregated notions are combined into the entities of the aggregate notion in all possible combinations. The inverse operation for aggregation is the decomposition operation, in which the aggregated notions are extracted from the aggregate notion.

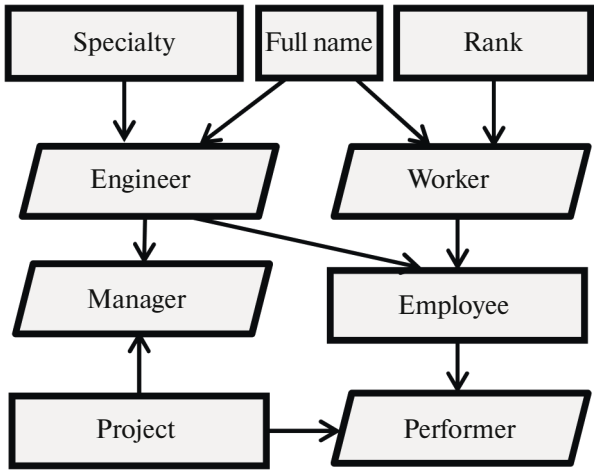


Fig. 1. Fragment of notional structure.

An example of an aggregate notion is the Space notion, which aggregates three independent Coordinate notions; the Complex Number notion, which aggregates two independent number notions; the Color notion, which aggregates three notions of basic colors (Red, Green, and Blue) of varying intensity.

It should be noted that any entity of the subject area can be considered in certain respects as a sign notion and in others as various abstract notions, i.e., it can be defined as generalizations or associations of other notions. For example, the Tree notion can be a generalization of tree species in one aspect and an association of roots, trunks, branches, leaves, etc. in another aspect.

As a result, we have the fact that the material given in the representations is transformed into notions using the following methods of logical-theoretical processing: operations of identification-interpretation, generalization–specialization (in particular, typification–concretization), and association–individualization (in particular, aggregation–decomposition) [17].

5. NOTIONAL MODEL

A notional model is a notional structure of a subject area with a description of the composition of the notions included in it. In a notional structure, notions are listed and the methods of their formation are indicated, i.e., intensions (contents of notions) are specified. The description of the composition of notions is carried out by specifying their extensions (scopes of notions), i.e., listing or resolving the sets of entities included in them.

To specify the intension of a generalization notion, curly brackets are used, in which the generalized notions are listed, and to specify the intension of an association notion, square brackets are used, in which the associated notions are listed:

$$\mathrm{Int}G = \{G1\ G2\ \ldots\ GK\}; \quad (1)$$

$$\mathrm{Int}\ A = [A1\ A2\ \ldots\ AL], \quad (2)$$

where Int G is the intension of a generalization notion with the name G; $G1$, $G2$, ..., GK are the names of the generalized notions; Int A is the intension of an association notion with the name A; $A1$, $A2$, ..., AL are the names of the associated notions; the equal sign is used to establish equivalence between the intension of the formed notion and the expression that defines it. Traditionally, curly brackets indicate an unordered set without repetition of elements, and square brackets indicate an ordered set with repetition of elements.

Figure 1 shows a fragment of the notional structure, which lists the Specialty, Full Name, Rank, Employee, and Project as generalization notions (shown as rectangles) and the Engineer, Worker, Manager, and Performer as association notions (shown as parallelograms). To simplify the example, the aspects of the notions are not indicated in the diagram. The notions that are not defined in the diagram are Specialty, Full Name, Category, and Project. The formation of the remaining notions is shown by arrows. Thus, the Employee notion is a generalization of the Engineer and Worker notions, and the Manager notion is an association of the Project and Engineer notions. The description of the volume of notions is carried out by specifying the enumerable or resolvable sets of entities belonging to them. Enumerable sets of entities are used to describe small-volume notions, and resolvable sets are used for large-volume notions.

For example, the Rank generalization notion is a type notion generalizing the categories of workers—sign notions 1, 2, and 3. In turn, the Full name generalization notion is a typification of strings consisting of the last names, first names, and patronymics of employees. The Specialty type notion is described similarly by listing the engineering specialties of the problem area. All these notions are described by enumerable sets of entities. However, the Full name and Specialty notions are specializations of the String notion, the set of entities of which is large and is specified by the corresponding resolution procedure.

Formulas (1) and (2) indirectly specify the volumes of notions. Thus, the generalization notion G from (1) consists of all entities belonging to the notions $G1$, $G2$, ..., GK,

$$\mathrm{Ext}G = \mathrm{Ext}G1 \cup \mathrm{Ext}G2 \cup \ldots \cup \mathrm{Ext}GK, \quad (3)$$

while the association notion A from (2) consists of entities obtained by combining the entities of notions $A1$, $A2$, ..., AL into the entity of notion A:

$$\mathrm{Ext}A \subseteq \mathrm{Ext}A1 \times \mathrm{Ext}A2 \times \ldots \times \mathrm{Ext}AL, \quad (4)$$

where Ext X is the extension of notion X; $\subseteq$ is the relation of nonstrict inclusion of sets; $\cup$ is the operation of union of sets; $\times$ is the operation of Cartesian product of sets.

Formula (3) describes the extension of the generalization notion, and formula (4) specifies the condition imposed on the extension of the association notion. If the equality of the left and right parts is true in formula (4), then we have a special case of association—aggregation of notions.

To determine whether an entity E belongs to an association notion A, the following formula is used:

$$A\{E\} \leftrightarrow \varphi(E, A1, A2, \ldots, AL), \quad (5)$$

where φ is a certain logical statement with the inclusion of free variables $E, A1, A2, \ldots, AL$, which limits the occurrence of entities of associated notions $A1, A2, \ldots, AL$ in the entity of association notion A; $\leftrightarrow$ is the logical connective "equivalent".

Formula (5) is constructed according to the following rules:

(1) The predicate of belonging of entity E to notion N of the form $N\{E\}$ is a formula (equivalent to $E \in \text{Ext } N$);

(2) A relation between entities of the form $(X[i] \bullet E)$ is a formula, where $X[i]$ is a functor returning the entity of the notion with number i ($i > 0$) from the entity X, $\bullet$ is a symbol of the relation that is defined for entities $X[i]$ and E (the relation "equal to" is used for any entities; the relations "greater than" and "less than" are used for ordered entities);

(3) An expression with the logical connective "not" of the form $(\neg \varphi)$ is a formula if φ is a formula;

(4) An expression with the logical connectives "and", "or", "implication" and "is equivalent" of the form $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$, and $(\varphi \leftrightarrow \psi)$ are formulas if φ and ψ are formulas;

(5) Expressions with a universal quantifier and an existential quantifier of the form $(\forall X\, \varphi)$ and $(\exists X\, \varphi)$ are formulas if φ is a formula and X is a free variable in the formula φ;

(6) Brackets in formulas can be omitted taking into account the following order of interpretation of relations, logical connectives, and quantifiers: $\bullet$, $\neg$, $\wedge$, $\vee$, $\forall$, $\exists$, $\rightarrow$, $\leftrightarrow$.

An example of a description of the extension of the Engineer association notion from the demonstration notional model (see Fig. 1) can be the logical statement:

$$\begin{aligned} &\text{Engineer}\{E\} \leftrightarrow (E[\text{FULL NAME}] \\ &\quad = \text{"Ivanov Petr Sidorovich"} \\ &\vee\ E[\text{FULL NAME}] = \text{"Petrov Ivan Ilyich"}) \\ &\wedge\ E[\text{Specialty}] = \text{"Mechanical} \\ &\quad \text{engineering technology".} \end{aligned}$$

It asserts that the composite entity E is an Engineer if and only if its part is FULL NAME or "Ivanov Petr Sidorovich" or "Petrov Ivan Stepanovich" and its part Specialty is "Mechanical engineering technology".

Table 1. Tabular form of the Engineer notion

Full name	Specialty
Ivanov Petr Sidorovich	Mechanical engineering technology
Petrov Ivan Ilyich	Mechanical engineering technology

Table 2. Table of the Engineer notion

Engineer	Full Name	Specialty
Engineer-1	Full Name-1	Specialty-1
Engineer-2	Full Name-2	Specialty-1

In addition to the formulaic description of entities belonging to notions, their tabular enumeration is used. In this case, the table header specifies the intension of the notion (its content), and the table body specifies the extension (its volume). For example, for the Engineer notion described above, Table 1 can be used. However, to prevent duplication of volumetric entities, their inclusion in other entities is replaced with short names. To match an entity and its name, a unique key field with the name of the notion is provided in the notion tables (Table 2). Similarly, the tables of Full Name and Specialty notions are created, in which entities with the names Full Name-1, Full Name-2, Specialty-1, etc. are defined.

The universal relation defined for all entities and notions is their equality. Two entities are equal if and only if they combine the same entities. For example, the Engineer-1 and Engineer-2 entities are not equal, since the Full Name entities that comprise them are different. The equality of notions is defined similarly: two notions are equal if and only if they consist of the same entities, i.e., when their extensions are equal.

The logical formulas used to express extensions are the values of the pure monadic predicate calculus [18, Theorem 5]. It is known that the pure monadic predicate calculus has many useful properties that make it similar to the propositional calculus [19, p. 24]: consistency, completeness on finite models, decidability, low expressiveness, and effective computability.

For logical models, attempts have also been made to limit the expressiveness of the logical language by using description logics. Description logics are built on pure monadic predicates of belonging of an instance to a concept and dyadic predicates of roles (relations between instances). To reduce description logic to a pure monadic predicate calculus, the use of roles is limited by the constructions $\forall R.C$ and $\exists R.C$, which reduce the application of dyadic predicates to pure monadic ones, linking the second variable of the role R with instances of the concept C [7, pp. 50, 150]. This made it possible to achieve the solvability of formulas and relatively low computational complexity of

solving the main problems of knowledge representation and processing in descriptive logics.

In notional models, it was possible to further reduce the expressiveness of the logical language by abandoning the use of dyadic subject predicates (roles) and to achieve greater efficiency in knowledge processing compared to descriptive models.

6. INFERENCE ON KNOWLEDGE

The use of objectified knowledge implies the presence of forms of knowledge representation and methods of manipulating them to imitate human reasoning. Usually, facts (judgments) are used to represent knowledge, and inference rules are used to process knowledge, allowing one to draw conclusions and obtain new statements (facts) based on the available facts.

In a notional model, generalization notions model the notional composition of knowledge in a subject area, and the intensions of generalization notions define the rules of inference according to which an entity E belongs to a generalization notion G if and only if it belongs to at least one generalized notion $G1$, $G2$, ..., GK:

$$G\{E\} \leftrightarrow G1\{E\} \vee G2\{E\} \vee \ldots \vee GK\{E\}. \quad (6)$$

In turn, association notions formalize judgments regarding notions, and the intensions and extensions of associated notions define the rules of inference according to which an entity E belongs to an association notion A if and only if it combines entities $E[1]$, $E[2]$,..., $E[L]$ of associated notions $A1, A2, \ldots, AL$, and the set of these entities satisfies the conditions specified by the logical expression φ:

$$A\{E\} \leftrightarrow A1\{E[1]\} \wedge A2\{E[2]\} \wedge \ldots \wedge AL\{E[L]\} \wedge \varphi(E, A1, A2, \ldots, AL). \quad (7)$$

The first part of the conditions of formula (7) specifies the belonging of entities $E[1]$, $E[2]$,..., $E[L]$ from the composition of the entity E being checked to the corresponding associated notions $A1, A2, \ldots, AL$ of the association notion A, and the second part of the conditions limits the arbitrary occurrences of entities of the associated notions $A1, A2, \ldots, AL$ in the entity of the association notion A.

A fragment of the notional structure, the diagram of which is shown in Fig. 1, generates the following inference rules, but not all:

$$\text{Employee}\{E\} \leftrightarrow \text{Engineer}\{E\} \vee \text{Worker}\{E\};$$

$$\text{Manager}\{E\} \leftrightarrow \text{Engineer}\{E[1]\} \wedge \text{Project}\{E[2]\} \wedge \varphi(E, \text{Engineer}, \text{Project}),$$

where φ is a formula that restricts the inclusion of pairs of entities of the type Engineer X of the Engineer notion and Project Y of the Project notion in the extension of the Manager notion, for example like this:

$$E[1] = \text{Engineer-1} \wedge (E[2] = \text{Project-1} \vee E[2] = \text{Project-3}) \vee E[1] = \text{Engineer-2} \wedge E[2] = \text{Project-2}.$$

When making inferences based on knowledge, it is necessary to establish or refute a certain fact by searching for entities that satisfy the specified conditions, for which queries to an understandable model are used. In this case, the inference rules of the type (6) define the generalized notions to be processed, and the inference rules of the type (7) allow us to establish the constituent entities of the processed notion, as well as to check whether an arbitrary entity belongs to the processed notion.

Using the names of entities and their variables, queries to the notional model are formulated, for example, this one: "Which of the performers of project A is a worker without the third rank?" In the language of the pure monadic predicate calculus, this query will have the following expression:

$$\text{Performer}\{X\} \wedge X[\text{Project}] = \text{Project-1} \wedge \forall Y(\text{Worker}\{Y\} \wedge X[\text{Employee}] = Y \wedge \neg(Y[\text{Rank}] = 3)),$$

where X is a free variable and Y is a bound variable. For clarity, the numbers of the notions of the constituent entities in the expression are replaced by their names: X[Project], X[Employee], and Y[Rank]. The query has a depth of 2 since a nested subquery with variable Y is processed for each entity X.

The execution time of simple queries to the notional model has a linear or logarithmic estimate, since searching for an entity in a notion table with N records is performed in time with an asymptotic estimate of N [20, p. 667], and in the case of using an index, in time with an asymptotic estimate of $\log N$ [20, p. 601]. If complex nested queries with a depth D and an average number of records in the viewed tables N are used, then the execution of a query to the notional model is performed in time with an asymptotic estimate of N^D or $\log^D N$, depending on the absence or presence of indexes in the notion tables.

7. PROCEDURAL KNOWLEDGE

So far, we have considered the representation of declarative knowledge, which is defined by various methods of forming notions and listing entities in their extensions. However, in addition to declarative knowledge, another type of knowledge is used in practice—procedural knowledge.

In artificial intelligence, procedural knowledge is a type of knowledge that can be possessed by an intelligent agent that perceives his environment and acts rationally to achieve its goals and acquires new knowledge in the process of its activity [15, p. 66]. Such knowledge is represented in the form of a finite

automaton or a computer program that implements the agent's function, mapping its perception of the environment to actions that affect this environment and change his perception.

In the theoretical aspect, procedural knowledge in the notional model is expressed in the language of the pure monadic predicate calculus and is used to define decidable sets of entities. In the applied aspect, procedural knowledge is represented as named sequences of actions (procedures) that allow one to obtain resulting entities and notions from some initial ones.

Procedures, like everything in notional models, are represented by entities of the corresponding notions. Elementary actions from which a procedure is built are the operations of creating, changing, and deleting entities of notions. Moreover, the notions themselves will be considered as entities of some notion of notions, which allows creating, changing, and deleting the notions themselves [21].

To increase the expressiveness of the language of knowledge representation and processing, we provide an assignment operator, as well as a conditional operator and cycle operators. The conditions of the operators are specified by logical expressions, the truth of which is checked when the operator is executed.

A feature of the namespace of the described language of knowledge representation and processing is the interpretation of all variables as notions. Any variable is a one-dimensional array of entities. If the array has one element, then such a variable is an entity, but at the same time it is a notion with one entity belonging to it.

To encode procedural knowledge, we will introduce a procedural data type, the entities of which are sequences of commands of a certain virtual machine that performs operations on entities of notions. Procedural knowledge is syntactically described as expressions of the form:

$$(X,Y,\ldots,Z) \Rightarrow W, \tag{8}$$

where X, Y, ..., Z are formal argument notions, and W is the procedure text in the knowledge representation and processing language with formal argument notions included in it.

The result of executing the procedure (8) is the return of some resulting notion, i.e., any procedure can be launched for execution by passing it the initial notions, and a result can be obtained, including a procedural one. A side effect of executing the procedure can be a change in the notional model used.

The built-in operations of the knowledge representation and processing language are operations on simple data types, namely: numbers of limited bit depth, dates in various formats, finite-length strings, binary data, procedures, etc.

As an example, let us consider a procedure that increases the ranks of workers participating in Project-1 (program 1).

Program 1. Defining and calling an anonymous procedure with the argument Project

```
(Project) => {
    // Defining the internal rank promotion function
    Promotion = (Rank) => Rank < 3 ? 1 : 0;
    // Creating an index variable and initializing it
    Index = 0;
    // Opening a transaction named Ranks
    begin Ranks;
    // Querying entities X of the Workers temporary notion
    Workers = {X: Performer} (X[Project] == Project && Worker{X[Employee]});
    // Looping through entities of the Workers temporary notion
    while (Index < Workers{})
    {
        // Creating a Worker local entity
        Worker = Workers{Index};
        // Incrementing the Rank attribute of the Worker entity
        Worker[Rank] += Promotion(Worker[Rank]);
        // Incrementing the index of the entity of the Workers notion
        Index = Index + 1;
    }
    // Updating changed entities in the notional model
    update Workers;
    // Successful completion of the transaction
    commit Ranks;
}(Project-1);
```

In the program text, a construction of the form $Y\{X\}$ is a functor for extracting an entity with the name (number) X from the extension of the notion Y. In a logical context, $Y\{X\}$ is a predicate of belonging of an entity X to the notion Y. If the argument is omitted in $Y\{X\}$, then $Y\{\}$ is the number of entities in the extension of the notion Y.

It should be noted that assignment operators create only temporary variables. However, the connection with the notional model is preserved for all entities extracted from it, which allows changes to be made to the notional model using the ***update*** operator. The language also contains operators for creating and deleting entities of notions.

8. LEARNING NOTIONAL MODELS

The supervised learning of notional models is a process in which the user (teacher) defines notions and lists the entities belonging to them, i.e., he specifies the intensions and extensions of notions. Similar approaches are used to fill other knowledge models with subject content.

However, notional models can also be effectively learned without a teacher. Unsupervised learning of notional models is carried out by the notional model "perceiving" the "surrounding world" in the form of a stream of entities entering the input and solving the problem of recognizing the notions of these entities, taking into account the knowledge stored in the notional model.

To reproduce the existing knowledge, we will construct a formal grammar $F = [N, M, T, P]$, generating entities of a given notional model, where N is a finite set of basic nonterminal symbols that coincide with the names of notions; M is a finite set of auxiliary nonterminal symbols that are the names of entities; T is a finite set of terminal symbols used to express entities; P is a finite set of productions of the form $S \Rightarrow \omega$; S is a nonterminal symbol of the grammar, $S \in N \cup M$; ω is a string of finite length consisting of terminal and nonterminal symbols; $\Rightarrow$ is the sign of the grammatical inference operation separating the left and right parts of the productions.

The inference rules (5), which define the belonging of the entity E to the association notion A, generate productions of the form

$$A \Rightarrow M1 \circ M2 \circ \ldots \circ ML, \tag{9}$$

such that $\varphi(E, A1, A2, \ldots, AL)$, where A is a nonterminal symbol of the grammar; $M1, M2, \ldots, ML$ are auxiliary nonterminal symbols, names of entities of the associated notions $A1, A2, \ldots, AL$ from the entity E; $\circ$ is the concatenation symbol separating the names of entities.

Productions (9) generate a set of strings that express entities E of the association notion A. Auxiliary nonterminal symbols M are used in entity naming productions:

$$E \Rightarrow \mu, \tag{10}$$

where E is the name of the entity $E \in M$; μ is its expression in the combined alphabet of terminal and nonterminal symbols of the grammar.

Let us give an example of a formal grammar for a demonstration notional model (see Fig. 1). The belonging of the entities of the notional model to the association notions is specified by the following productions of the form (9):

Full name ⇒ Full name-1 | Full name-2 | Full name-3 | Full name-4;

Specialty ⇒ Specialty-1 | Specialty-2;

Rank ⇒ Rank-1 | Rank-2 | Rank-3;

Project ⇒ Project-1 | Project-2;

Worker ⇒ Full name-3° Rank-3 | Full name-4 ° Rank-1;

Engineer ⇒ Full name-1 ° Specialty-1 | Full name-2 ° Specialty-2;

Contractor ⇒ Worker-1 ° Project-1 | Worker-2 ° Project-2;

Manager ⇒ Engineer-2 ° Project-1 | Engineer-1 ° Project-2,

where | is a symbol separating alternative options in the right part of the product, and auxiliary nonterminal symbols (10) are expressed as follows:

Full name-1 ⇒ "Ivanov Petr Sidorovich";

Full name-2 ⇒ "Petrov Ivan Ilyich";

Full name-3 ⇒ "Sidorov Ilya Ivanovich";

Full name-4 ⇒ "Ilyin Sidor Petrovich";

Specialty-1 ⇒ "Mechanical engineering technology";

Specialty-2 ⇒ "Injection molding";

Project-1 ⇒ "Project A";

Project-2 ⇒ "Project B";

Rank-1 ⇒ 1;

Rank-2 ⇒ 2;

Rank-3 ⇒ 3.

A formal grammar with productions of the form (9) and (10) is context-free. For context-free grammars, the problem of determining whether an arbitrary string belongs to any nonterminal symbol of the grammar is solvable [22, p. 352].

It follows that all entities of the notional model can be recognized and their concepts can be defined. If the notion of some entity from the input stream is not recognized, then unsupervised learning solves the problem of replenishing the extensions of the corresponding notions with new entities, in which the input entity is recognized. If the entity cannot be recognized by replenishing the extensions, then a new notion, the extension of which includes the unrecognized entity, is created.

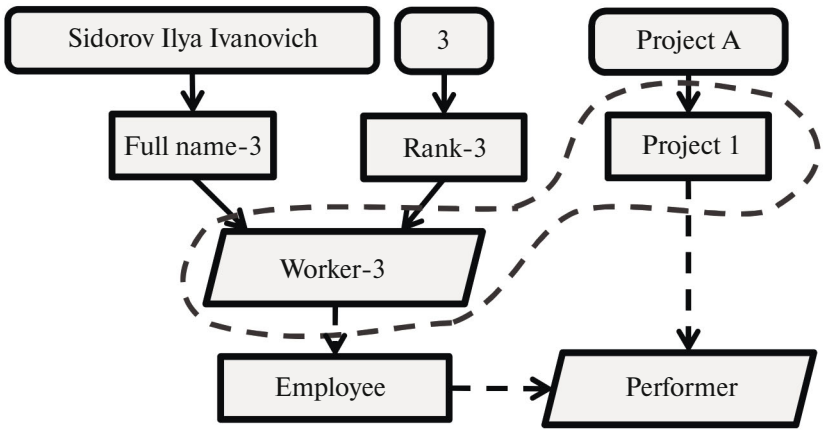


Fig. 2. Grammatical parsing tree.

New entities of the notional model must be derived from the productions of the extended grammar F following from the inference rules (6) and (7):

$$G \Rightarrow G1|G2|\ldots|GK, \quad (11)$$

$$A \Rightarrow A1 \circ A2 \circ \ldots \circ AL, \quad (12)$$

where G is a nonterminal symbol of a generalization notion; $G1$, $G2$, ..., GK are nonterminal symbols of generalized notions; A is a nonterminal symbol of an association notion; $A1$, $A2$, ..., AL are nonterminal symbols of associated notions.

Productions (11) generate a set of strings G that express the entities of a generalization notion if the sets of strings $G1$, $G2$, ..., GK that express the entities of the corresponding generalized notions are known. In turn, productions (12) generate sets of strings A that can express the entities of an association notion if the sets of strings $A1$, $A2$, ..., AL that express the entities of the corresponding associated notions are known.

Let us consider an example of unsupervised learning of a notional model when the following training data is received: "Sidorov Ilya Ivanovich" 3 "Project A." Figure 2 shows the tree of grammatical parsing of the input string, from which it follows that the entity Worker 3 ° Project 1 is not recognized (circled with a dotted line).

Then, to recognize new entities, grammatical parsing continues in the grammar expanded by the following productions:

Engineer ⇒ Specialty ° Full name;

Worker ⇒ Full name ° Rank;

Manager ⇒ Engineer ° Project;

Employee ⇒ Worker | Engineer;

Performer ⇒ Employee ° Project.

From the last two productions, the unrecognized entity Worker 3 ° Project 1 is derived, which is included in the extension of the Performer notion (dashed arrows in Fig. 2), and to recognize the added entity, the production Performer ⇒ Worker 3 ° Project 1 is added to the grammar.

With ambiguous grammars expressing entities with alternative interpretations, several variants of grammatical parsing of training data may be encountered. In this case, each of them is checked and the one that introduces, for example, the smallest changes to the notional model is selected.

CONCLUSIONS

The knowledge model is called notional to distinguish it from conceptual models. A notion differs from a concept. A concept is objective, while a notion is subjective. For this reason, there are many formal notions that have the same name but different volumes and contents in different problem areas (aspects). In this case, a concept can be defined as a result of objectification of notions of the same name in different aspects.

Using the operations of identification, association, and generalization, all conceivable notions can be constructively constructed, and the belonging of enumerable or decidable sets of entities to the constructed notions can be specified. In a notional model, the extensions of notions are described in the language of pure monadic predicate calculus, which makes notional models decidable and consistent in countable subject areas and complete in finite ones [18]. By reducing the expressiveness of the notional language, the computational complexity of knowledge representation and processing problems is reduced compared to other knowledge models.

The difficulties of knowledge representation and processing in logical, production, network, and frame models do not allow the effective implementation of large knowledge bases. However, when the notional

model of knowledge was used, it was possible to obtain some advantages:

– low computational complexity of inference on knowledge, no need to use heuristics;

– queries to the notional model are executed in polynomial time from the logarithm of the average number of entities in the extensions of notions, and the query results have a strict logical justification and a meaningful explanation;

– the ability to narrow the area of knowledge by using notions in various aspects;

– the use of procedural knowledge formalized in a developed procedural language with operations for creating, changing, comparing, and deleting entities of notions, where they are also entities of a certain notion—the notion of notions;

– unsupervised learning of notional models by grammatical parsing of the flow of input sign notions, identifying unknown entities in it to replenish the extensions of existing notions with new entities or to form new notions;

– the clarity of the notional model due to its direct reproduction of ideas, notions, and judgments of the subject area of knowledge.

The high efficiency and expressiveness of notional models and their easy updating, which does not lead to the emergence of hidden contradictions, allows us to hope for the widespread introduction of notional representation of knowledge into the practice of creating and using modern intelligent information systems.

It should also be noted that there are no other more general expressive means for representing knowledge than their description in the form of a notional model [23]. This is explained by the fact that notional thinking is an extremely general and the only mechanism of rational cognition, and notional analysis is a subjective and, ultimately, objective form of recording the results of notional thinking.

However, possession of knowledge and the ability to think (to make a conclusion on knowledge) is not sufficient for strong artificial general intelligence. There are many problems that cannot be solved by rational methods, such as the synthesis and analysis of algorithms and programs, proving theorems, large-scale combinatorial problems, global optimization, etc. To implement strong artificial general intelligence, something more is needed, such as the use of not only rational but also irrational forms of cognition, as well as emotions.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The author of this work declares that he has no conflicts of interest.

REFERENCES

1. *Handbook of Knowledge Representation*, Harmelen, F., Lifschitz, V., and Porter, B., Eds., Elsevier, 2008.
2. Chein, M. and Mugnier, M.-L., *Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs*, Advanced Information and Knowledge Processing, London: Springer, 2009. https://doi.org/10.1007/978-1-84800-286-9
3. Gough, B. and Madill, A., Subjectivity in psychological science: From problem to prospect., *Psychol. Methods*, 2012, vol. 17, no. 3, pp. 374–384. https://doi.org/10.1037/a0029313
4. Polya, G., *Mathematics and Plausible Reasoning*, vol. 2: *Patterns of Plausible Inference*, Princeton, NJ: Princeton University Press, 1954. https://doi.org/10.1515/9780691218304
5. Walton, D., *Abductive Reasoning*, Tuscaloosa, AL: University of Alabama Press, 2005. https://doi.org/10.2307/jj.30297486
6. Brachman, R.J. and Levesque, Y.J., *Knowledge Representation and Reasoning*, Elsevier, 2014.
7. *The Description Logic Handbook: Theory, Implementation and Applications*, Baader, F., Calvanese, D., McGuinness, D.L., et al., Eds., New York: Cambridge University Press, 2003.
8. Klahr, D., Langley, P., and Neches, R., *Production System Models of Learning and Development*, Cambridge, MA: MIT Press, 1987. https://doi.org/10.7551/mitpress/5605.001.0001
9. Sowa, J.F., *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Thomson Learning, 2000.
10. *Object-Oriented Design Knowledge: Principles, Heuristics, and Best Practices*, Garzas, J. and Piattini, M., Eds., Idea Group Publishing, 2007.
11. Gardarin, G. and Valduriez, P., *Relational Databases and Knowledge Bases*, Addison-Wesley, 1989.
12. Vykhovanets, V.S., The notional model of knowledge representation, *J. Phys.: Conf. Ser.*, 2021, vol. 1864, no. 1, p. 12058. https://doi.org/10.1088/1742-6596/1864/1/012058
13. *The Oxford Dictionary of Current English*, Thompson, D., Ed., New York: Oxford University Press, 1999.
14. Olivé, A., *Conceptual Modeling of Information Systems*, New York: Springer, 2007. https://doi.org/10.1007/978-3-540-39390-0
15. Russell, S.J. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Hoboken, NJ: Pearson, 2022.
16. Vykhovanets, V.S., About essential incompleteness of a formal method, *Filosofiya, matematika, lingvistika. Aspekty vzaimodeistviya. Materialy Mezhdunarodnoi nauchnoi konferentsii* (Philosophy, Mathematics, Linguistics: Aspects of Interaction: Proceedings of the International Scientific Conference), St. Petersburg, 2009, St. Petersburg: VVM, 2009, pp. 234–240.

17. Vykhovanets, V.S., Intelligent information systems based on notional models without relationships, *Advances in Intelligent Systems, Computer Science and Digital Economics II. CSDEIS 2020*, Hu, Z., Petoukhov, S., and He, M., Eds., Advances in Intelligent Systems and Computing, vol. 1402, Cham: Springer, 2021, pp. 20–28. https://doi.org/10.1007/978-3-030-80478-7_3
18. Vykhovanets, V.S., The notional analysis and notional modeling, *Upravlenie Bol'shimi Sistemami. Sbornik Trudov*, 2021, no. 92, pp. 64–109. https://doi.org/10.25728/ubs.2021.92.4
19. Mendelson, E., *Introduction to Mathematical Logic*, New York: Chapman and Hall/CRC Press, 2010.
20. Elmasri, R. and Navathe, S.B., *Fundamentals of Database Systems*, Hoboken, NJ: Pearson, 2016.
21. Vykhovanets, V.S., The knowledge description language, *14th International Conference Management of Large-Scale System Development (MLSD)*, Moscow, 2021, pp. 1–5. https://doi.org/10.1109/MLSD52249.2021.9600103
22. Reghizzi, S.C., Breveglieri, L., and Morzenti, A., *Formal Languages and Compilation*, Texts in Computer Science, Cham: Springer, 2019, 3rd ed. https://doi.org/10.1007/978-3-030-04879-2
23. Vykhovanets, V.S., The methodology of notional analysis of complex subject domains, *Naukoemkie Tekhnologii*, 2022, vol. 23, no. 8, pp. 60–68. https://doi.org/10.18127/j19998465-202208-07

Publisher's Note. Allerton Press remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
AI tools may have been used in the translation or editing of this article.