Chapter

Computational Model for the Construction of Cognitive Maps

Larisa Yu. Ismailova, Sergey V. Kosikov and Viacheslav E. Wolfengagen

Abstract

The chapter considers an option for solving the problem of storing data in the Web environment and providing an access to the data, taking into account their semantics, i.e., in accordance with the nature of the tasks solved by users of different classes. The proposed solution is based on the use of presentation of the data in the form of semantic networks. As the main technical tool for describing access methods, the chapter proposes cognitive maps (CMs), which can also be considered as semantic networks of special type. When access is done, the presentation of information consistent with the semantic description of the user is provided. The suggested method of constructing CMs is based on the intensional logic. The solution is presented in the form of a computational model, which provides for the construction of CM's dependence on the parameter. The proposed method of parametrization makes it possible to take into account the semantic characteristics of users of various classes. Some CM constructions for problem domain description are presented. A method for semantically oriented naming of CMs is proposed. The method is based on building of a functor of special type.

Keywords: cognitive maps, access method, semantic network, interpretation, computational model, intensional logic, functor

1. Introduction

As Web technologies progress the task of developing tools for the data organization and storage in a web environment assumes ever greater importance [1]. In the pre-network single-user environment, the prevailing way to organize data was a hierarchically organized file structure. Still this method was convenient for not all tasks (in particular, it did not directly provide the possibility of simultaneous classification of various information objects according to various foundations). However, in the whole, it covered a rather wide class of applications.

The situation changed thanks to network technologies, when information objects turned out to be associated with different users, who applied in general different principles for placing, searching, and processing. Thus, the rigidly defined hierarchies were replaced by network structures that determine the connections of information resources set by various ways. The connections may arise due to the different reasons, and this assumes the need, firstly, to process them in different ways and, secondly, to take into account the meaning of the data and their connections when processing. Thus, the data become essentially semantic in nature.

The semantic measurement of data structures leads to the necessity to change respectively the methods of their description, searching in the environment of such structures and manipulating them [2].

The need to use semantically rich network structures requires to develop semantically oriented methods for describing data structures and their processing, in particular, the definition of semantically oriented search methods [3]. Different users must work with different fragments of data, which are determined by the goals of their work, the source data that is necessary to achieve the goals, a set of appropriate access rights, etc. All this raises the problem of providing access to data, which should take into account both the class of the user, getting the access, and its characteristics, as well as the semantics of the data, to which the access is allowed, in particular, providing a user-friendly representation of the data.

The means of describing users and the means of their access to data, considering the specifics of the tasks to be solved, must combine power enough to distinguish the relevant elements of the description and simplicity. It makes it possible to practically use the descriptions without cumbersome instrumental kits. The means of description should fix the user's view on the subject area, which should be sufficiently detailed to describe the classes of problems to be solved, but without redundant detailing that might lead to an increase in the volume and complexity of the description. The use of cognitive maps (CMs) seems promising in this aspect.

Cognitive maps are diagrams used to visually organize information. Various works define the cognitive maps in various ways. For example, CM's can be used to represent spatial relations and determine the mutual position of information elements in a physical environment. CMs also can be used to represent abstract information and to map it to the spatial (usually planar) view. In this case CMs are connected with the presentation of information described as mind maps, which, thus, can be considered as a specific instance of CMs. As a rule, CMs represent information in the form of a hierarchical structure. It is also possible to represent connections between nonadjacent elements of the structure, which brings CMs closer to the network representation of information. The elements of the structure are connected by arcs, which, as a rule, are not specifically marked. The connections represented by arcs can use the order of homogeneous nodes that are on the same level of hierarchy. This allows to provide not only structural information but also, for example, information on the sequence of actions for solving any task.

Using CM to describe the characteristics of the user, including his typical tasks, allows to perform this description in the early stages of designing an information system. The study in the field of cognitive psychology allows us to characterize CM as a means of knowledge structuring, consistent with the human way of thinking in solving practical tasks. For this reason the CM can be used to fix ideas about the user and his tasks up to creating a formal domain model, moreover, as one of the tools for developing such a model.

Semantically oriented structures for data presentation are proposed for solving user's problems. It is worthwhile to choose the representing structure in the form of a semantic network [4]. The semantic network is understood as an oriented graph consisting of nodes and marked arcs. The nodes correspond to concepts, i.e., notions of various degrees of generality, presented in the network. The arcs correspond to the connections of the concepts among themselves, and the marks of arcs determine the way of interpreting the connections. As a rule, arcs are lined not randomly but in accordance with certain patterns representing stereotypical sets of connections (frames).

The semantic network may be of a tree-structured nature or include tree-structured fragments in its composition, but it does not mean the network necessarily has this form. Therefore, the semantic network (SN) is actually a more general structure for the representation of knowledge about the domain than CM. Following

this, the description of the user and his view on the subject area using a cognitive map can be considered as a specific instance of the semantic network. The transition from CM to the description in the form of a semantic network can be performed using two basic operations: (1) recovering unclear marks on arcs in CM and (2) deriving semantic network configuration patterns corresponding to the CM fragment.

The interpretation of CM as a special type of semantic network at a conceptual level provides the inclusion of the user in the conceptual model of the domain as part of it. The works on semantically oriented data representation [1, 4] show that the model should include constructions that describe the relationship between the semantics of user characteristics and the semantics of the data being processed. Such model provides the user with data for processing in accordance with the set of his powers and the nature of the problem to be solved. When the model is represented as a whole in the form of a semantic network, its considered part can be represented as a control subnet, which provides the computation of query results in accordance with the specified parameters.

One of the critical tasks in organizing access and processing semantically oriented data is the preservation of semantics while working with resources [5]. The resources, on the one hand, exist for a long time, and this makes their repeated use possible, and, on the other hand, they usually have a dynamic nature, i.e., they can be modified, updated, etc., at any point in time. In this case the change is possible of both the data about separate facts, processes, etc. in the area described by the resource set and the general semantic characteristics of the data. The change is also possible of both the data itself and the links, i.e., the dynamic arising of new links, changing the semantics of existing links and other information objects.

The changes in semantics can both save the logical continuity of a network and violate it. This chapter understands the logical continuity as the preservation of a set of general constraints (including informal ones) placed on the contents of the network. Maintaining logical continuity during network modification requires dynamic checking of constraints when performing operations that change the semantics of the network in order to prevent actions that destroy the semantics.

For supporting the changes, the most important factor is that the semantics can be changed both unintentionally (by chance) during the work and maliciously. In the latter case, the goal may be to get unauthorized access to information—receiving or changing of the information. For this reason the support of the semantic integrity of the network suggests, in particular, restricting the user's access to information in the network. Such a restriction may include both usual restrictions on writing or reading and more complicated semantically motivated constructions. For example, the possibility to change data in the network only in a strictly determined way or every change must be associated with the information that identifies the user who made the changes.

The development of tools for supporting the network access operations involves describing the semantics of a system of interconnected resources by a formalized way, which makes it necessary to introduce the concept of a semantic network as a formal analogue of a resource system [6]. The capabilities of network access tools and their restrictions are shown in the form of a model of access tools to the SN. The necessity to implement tools for supporting access operations leads to the fact that the model must be of a computational nature. The development of such a model is expected to provide the ability to construct a semantically correct system of support for access to the SN, including the ability to specify semantically consistent access restrictions, which makes the task of developing a computational model for supporting access operations relevant. The need for data processing, including the designation of queries, suggests the development of a theoretical basis for constructing a computer information system that provides both the logical

correctness of data interpretation and the construction of appropriate computational procedures. For this purpose this chapter uses the formalism developed on the basis of intensional logic [6]. The computational aspect is ensured, in particular, by the possibility to include the means of a typical lambda calculus into the logical system under consideration.

The necessity to take into account the subjective view on the semantic network requires modeling the dependence of the interpretation of the system's structures on the subject. This requirement is considered in the intensional logic by defining an interpretation structure using a parameter, the assignment point. The value of each construction corresponds to specified parameter value. In this case the constructions of the language of intensional logic are divided into extensional and intensional ones. The value of the composite extensional construction at the specified assignment point is a function of the values of its constituent structures at the same assignment point. The determination of the value of the intensional composite constructions requires determining the values of its constituent constructions on the entire set of assignment points or on some of its subsets.

To take into account the interpretation of various entries of an information object requires the construction of models of interpretation dependence on the context. The context determination can also be performed using intensional constructions. In this case, it is possible to use intensional operators or constants—intensions of higher orders.

The applied method of parameterization allows to take into account the semantic characteristics of users of various classes.

This chapter is structured as follows. Section 2 describes some approaches to the definition and construction of cognitive maps; special attention is paid to the degree of use of semantic information. Section 3 contains a statement of the problem of supporting the language of description of cognitive maps and means of its interpretation and offers a solution as a variant of the language of intensional logic. Section 4 describes the use of cognitive maps in the description of the problem area on the example of dependent types. In Section 5 we propose an approach to build a support system of cognitive maps on the basis of adjoint functors. In conclusion, the results are summarized briefly.

2. Related work

The use of CMs for setting management methods of access to the distributed semantic network assumes the study of their expressive potential in the whole. In this connection approaches to formal definition of CM's semantics cause special interest. The diversity of approaches is reasoned not least by the fact that different researchers use the term "cognitive map" in various meanings.

The CM's applications cover different areas: sociology, economics, medicine, international relations, etc. Among the problems solved by CMs, the following ones can be singled out: (i) problems of conceptual modeling, especially in the context of initial understanding of problems in weakly structured subject areas; (ii) problems of further modeling of subject areas, especially if it is necessary to describe the dynamics; and (iii) management problems in the subject area. Some lines of research and application of CMs are shown in **Figure 1**.

Nevertheless, along with all the diversity of tools for creating CMs, including those widely disseminated ones, FreeMind [7], MindMeister [8], MindManager [9], Cacoo [10], MindMup [11], XMind [12] etc., only a small part of them can be considered as tool kits for cognitive modeling based on CMs or a ready-to-use tool for supporting cognitive architectures.

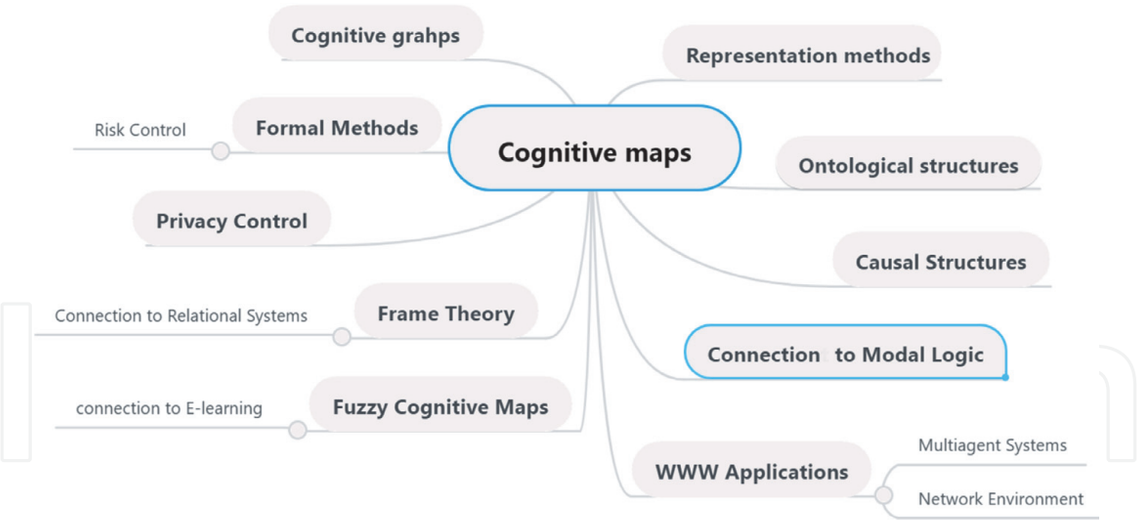

Figure 1.
Lines of research and application of CMs.

Regarding this some lines of research in the field of cognitive modeling should be mentioned. The work [13] proposes the most common approach. According to the approach, the cognitive modeling is "a line based on a knowledge-intensive interdisciplinary methodology for solving applied problems through cognitive maps with more or less support to special information technologies." In this case the cognitive map is understood as a formalized model of the situation that reflects the knowledge and/or beliefs of the subject, individual or collective, about the cause–effect impacts between the important factors of the situation.

Within the considered line, the formal models of CMs relate to the questions of reducing the risk, introduced by the human factor, when solving problems in various subject areas using CMs. The work [13] proposes an option of describing the approaches to the formal definition of the methods of interpreting CMs. This work distinguishes two approaches to the interpretation of CMs: descriptive (pinning methods of understanding the notions of the subject area) and normative (fixing the methods of solving problems in the subject area), which trace back to the approach accepted in the work [14]. The first approach aims to use CMs for developing an internal model of a man's knowledge about a certain situation. The second approach suggests CMs of different types as normative models (schemes or rules) for the external presentation of knowledge about situations.

In general, depending on the objective of the study, the details of the CM's definition differ from each other; in particular, CMs may have a different structure. In the whole within the formal approach, the CM definition is often extended to a cognitive graph. For example, the work [15] considers the structure of spatial knowledge that arises from the study of a new spatial environment and gives grounds for generalizing CMs up to cognitive graphs. The studies of the optimization of CM's representation adjoin the works of this type. Thus, the work [16] proposes a three-dimensional representation of CMs. The representation is based on the selection of the node kernels and daughter nodes, the nodes being located in three-dimensional space and being represented by balls of different radius. The proposed representation, as stated, enhances the cognitive clarity of the representation, which is interpreted as the ease of its intuitive understanding.

A cognitive map or, with a graph-based approach, a cognitive graph can represent parts of systems with a cognitive architecture and in this way be put in one or another cognitive architecture. The work [17] describes some cognitive architectures, the method of description giving an opportunity to think about the

compatibility of the presented architectures with the formalism of cognitive maps. The abovementioned work understands the cognitive architectures as software systems that might think about problems in different areas, develop ideas, adapt to new situations, and reflect upon themselves. To this end, the cognitive architectures are trying to provide evidence of which specific mechanisms successfully reproduce intellectual behavior and thereby contribute to cognitive science.

The considered paper emphasizes three large classes of cognitive architecture's character-coded, emergent, and hybrid ones. The character-coded systems represent concepts using characters that can be manipulated using a given set of operations. The emergent architecture assumes the use of multi-node parallel models, in which the flow of information is represented as the propagation of signals between the nodes. The hybrid architecture combines both approaches, but this combination can be made in different proportions. It is obvious that the system based on cognitive maps in this classification must be referred to the character-coded architecture.

The flexible way to represent data with different degrees of abstraction using CMs grounds the possibility of data using to represent ontological information. The work [18] considers how CMs can be used for the work in the situation when the information is missing or is unreliable in e-commerce. The paper presents a knowledge management system based on CMs and ontology and also proposes a framework solution for joint use of information along with the use of a common repository based on CMs. Using CMs provides modeling of a virtual environment by generating and checking the sequence of events that take place in the environment when modeling.

An interesting use of the CM's capabilities to represent dynamic information is the modeling of cause–effect (causal) relationships. Thus, the work [19] identifies the cognitive nature of a business model designated for a cognitive representation that describes business development activities. Attention is also drawn to the cause–effect structure of the business model, that is, the model of cause–effect relationships which, according to top managers or entrepreneurs' view, connects the creation of value and activities for its creation. The conceptualization and analysis of business models as cognitive maps can shed light on four important properties of the causal structure of a business model: levels of complexity, focusing and clustering, which characterize the causal structure, as well as the mechanisms underlying causal relationships shown in this structure.

There were some attempts to model CMs with the help of more general modal-logical contexts. The work [20] proposes an interpretation of cognitive maps, correlated with elements of large-scale spatial environment, for constructing geometrically impossible environments. Then the constructed CMs are proposed for joint interpretation with geometrically possible maps. Such an interpretation logically corresponds to the possibility of considering the interpreted cognitive map from different points of view, and the case of geometrically impossible interpretation is not excluded in advance.

The CM's use in the network environments (in particular, in WWW) is based on the CM's capabilities to represent information in a form that allows storing some nodes of map on separate nodes of the computational network, as well as parallel processing of stored information. Thus, for example, the work [21] is a description of the CM's use for working in a multi-agent environment. The strength of this work is the exact semantics of CMs, based on relational algebra according to [22]. Unfortunately, the constructed semantics has a very special form due to the chosen ad hoc three-valued logic system. Nevertheless, within the framework of the chosen semantics, it is possible to construct the forms for representation (when describing the subject area) of the agents' point of view on cognitive maps, as well as to determine decision-making procedures for such agents.

A somewhat different approach is adopted in the work [23], where cause–effect relationships are modeled based on interactive cognitive maps. A cognitive map is considered as a family of cognitive models. The models can be computed in parallel by exchanging data between themselves. In such conditions, the network implementation becomes natural, which also allows to hide data that a particular component "does not want" to make it visible to other components. The paper takes up the position that the adoption of the CM's network model leads to the construction of the CM's ecosystem, the development of which is managed by cognitive agents—system components.

Apart are fuzzy cognitive maps. This rapidly advancing branch, develops the formalisms of cognitive maps. In the general case, a fuzzy cognitive map is defined as a set of nodes and links, the nodes being associated with the concepts of the domain, and links to causal relationships between concepts. Each node is associated with the degree of the presence of a concept in a situation—a number or an element of a qualitative scale with which a number is associated. The nodes of the graph also associate with numbers that determine the degree of influence of one concept to another. A positive number corresponds to an increase in the presence of the corresponding concept and a negative one to its decrease.

The specified fuzzy cognitive map can serve to model the dynamics of a situation. To do this the initial degree of the concept's presence in the situation is set. Then the changes of the degrees of presence are determined in accordance with the links of the graph as the sum of the corresponding degrees of influence. The given process is repeated iteratively until it reaches the specified time limit. The experiments demonstrate that three main types of behavior are possible: (1) stabilization, i.e., convergence at a given point; (2) way out to the cyclic mode; and (3) chaotic behavior, characterized by the absence of limit modes.

An approach based on fuzzy cognitive maps is exampled in the work [24]. This paper shows the use of cognitive maps for making the decision, which is understood as the choice of a single decision or a group from the given set of alternatives. The cognitive maps are used thanks to their ability to explain the applied process of thinking. The work studies the process of convergence of cognitive maps and their application for decision-making.

The fuzzy cognitive maps can be used in different domains, including optimizing the learning process. One of the optimization techniques is to analyze data from learning management system logs and to identify patterns of users' behavior related to the content. The work [25] proposes the use of fuzzy cognitive maps to model the behavior of users of learning management systems. The proposed model describes the user's interaction with the content of the system and can be used to forecast the reaction of users to its training, test, and practical elements.

The relational approach to the construction of CM's semantics is gradually getting more of dissemination. So, besides the already cited work [21], the relational approach is also accepted in the work [26], in which dynamic models of fuzzy relational cognitive maps are analyzed. A frame-based approach, accepted, for example, in [27], can be considered as a generalization of the relational approach. In this case the frames are considered as stereotypical structures that provide orientation in the physical or conceptual space. In addition to the orientation, the choice of path can be provided, which corresponds to the solution of the planning problem. The frame approach is a synthesis of graph representations and cognitive maps and solves problems connected with explaining orientation-based behavior on graphs or maps or when they are used in parallel.

The frame approach can be successfully applied both in systems with common objectives and in systems oriented to specific applications. Thus, the work [28] solves the problem of presenting historical knowledge on the basis of CMs,

practically, on the basis of the frame approach. Actually, the CM's models are characterized as a specific type of dialectic interaction of logical and graphic forms of knowledge representation.

The considered work contains a detailed classification of cognitive maps. Thus, depending on the construction technology, they distinguish (1) associative maps or mind maps based on associations and (2) conceptual maps that serve to represent the connections of concepts between them. Among the mind maps are the maps identified as follows:

1. Dyadic, containing two alternative branches
2. Complex (poly-categorial), the number of branches in which is not limited (in practice it is convenient to have from three to seven branches)
3. Mnemonic, used to create an easily remembered image
4. Creative
5. Collective (e.g., developed during the implementation of joint creative projects)
6. Artistic

It is easy to see that the classification is based on various reasons, which makes it possible to set the task of clarifying the classification of CMs both for cognitive modeling and developing the formalizations oriented to their analysis, processing, and software generation.

All described applications may be characterized by one common feature—they are either not based on the use of formal semantics and use CMs as a convenient representation of knowledge about the subject area for informal analysis or, at best, use CMs as a tool for determining a finite state machine of a special type. However, such an approach seems to unreasonably narrow the scope of CM's application. It seems more reasonable to consider cognitive maps as the formalism, providing, on the one hand, pinning informal considerations about the described subject area and, on the other hand, obtaining more or less formalized descriptions that are compatible with descriptions in modeling languages or even programming languages.

An important sphere of application of solutions based on cognitive maps is information support for legal applications. For example, the work [29] analyzes the findings and contributions of existing research in the field of decision-making about the confidentiality, and it proposes to fill up the gaps in the modern understanding by applying a cognitive architecture to model confidential decision-making. In order to solve the issues related to confidentiality, it is necessary to consider aspects of human cognition, using, for example, the methods used in human-computer interaction and computer science research.

3. Intensional language for CM's description

3.1 Intensional CMs

An essential characteristic of the semantic network of the species considered in this chapter is the possibility of linking the structures corresponding to CMs with logical formulas of a certain kind. Because the meaning of the CMs depend

essentially on time, subject, etc., the logic appropriate for the basis for the interpretation of CMs must be explicitly focused on the consideration of semantic factors. Intensional logic can be chosen as such logic.

The intensional logic allows to operate with the formulas containing functional abstraction and application of function to arguments. Thus, it is possible to obtain the value of CM's structures using the evaluation. The result of the computation can also be represented as a CM's construction. In this case, the value depends on the parameter—the assignment point—which gives the CMs an intensional character.

The need for an intensional description of CMs leads to the problem of determining the language means of parameterized computation of semantic network structures as the task of developing methods to support a specialized language for describing the semantic network and means of its interpretation, which should provide:

1. The definition of means of interpretation of CM's structures on the basis of their assigned semantic characteristics

2. The definition of interpretation methods as specialized CMs, which can be embedded in objects that parameterize the interpretation

3. The definition of general limitations on interpretation methods, as well as procedures for the harmonization of interpretations that ensure the implementation of the imposed restrictions

The solution of the problem is supposed to be obtained on the basis of a combination of methods of intensional logic to describe the language and applicative methods of interpretation to compute the values of CMs. At the same time, it is possible to describe some constructions of the domain model in the form of CMs. The chapter presents a description technique on the example of dependent types.

Support to the implementation of intensional descriptions CMs requires the use of methods which agree with the methods of the description of the CMs. In this chapter, a functor technique is used for this purpose. The specialized functors are determined to represent CMs in supporting the programming environment. The definition is based on the adjoint functors.

The research method centers on the systematic use of the formalization of CM with the further determination of the semantics of the constructed formal objects. The object formalization is carried out using methods of intensional logic by constructing an intensional language to describe the objects that compose the CM. The intensional nature of the language makes it possible to take into account the contexts of objects used. The means of intensional logic provide for both the definition of objects, the interpretation of which is independent of the context (extensional objects) and objects of a different kind, and the interpretation of which requires consideration of one or more contexts (intensional objects). The intensional operators serve as the tools for setting contexts.

The semantics of objects is determined by the means of category theory. The use of category theory ensures a sufficient general definition of semantics, on the basis of which types of changes in the domain can be taken into account. Changes, in particular, can affect the domains of change of the variables of the CM description language, forming the so-called variable domains. Taking into account the changes allows describing the dynamic subject areas of the same CM, which in practical terms saves the efforts spent on developing and debugging the descriptions of CM use.

The analysis of methods of CM use to describe the subject areas consists of systematic consideration of the applied formalized methods and the identification of stereotypical structures used to describe objects and situations specific to a

particular domain. Considerable interest is caused by the study of general categorical constructions, such as functors and natural transformations, in their application to CM. In particular, the adjoint functor construction can be used to describe abstract types of the data associated with CM nodes.

3.2 Intensional language

The intensional language contains tools for describing the nodes and links of CM. The description in the intensional language is a formalized object, matched to CM. Such an object can be used both for constructing the semantics of CM and (in practical terms) for representing CM for the purpose of storage and processing. The use of formalized objects also provides for syntactic and semantic control of objects, which makes easier their debugging and maintenance.

Types are assigned to the expressions of the intensional language; thus, the type e corresponds to the node of CM and the type t to the link between specific nodes. A set of language expressions is defined as an inductive class. This method of setting ensures the definition of CM construction operations from separate parts. The description of the language as a whole follows the paper [6]. The interpretation of language expressions is also set with the help of induction for the construction of an interpreted expression.

3.2.1 Types

The set of types of Υ is defined inductively:

1. $e \in \Upsilon$, which is interpreted as the entity type.

2. $t \in \Upsilon$, which is interpreted as the type of sentence.

3. If $a, b \in \Upsilon$, then $\langle a, b\rangle \in \Upsilon$ and $\langle s, a\rangle \in \Upsilon$, where s is interpreted as the type of meaning.

4. There are no other types.

Types represent the sets of elements to interpret CMs or their fragments.

3.2.2 Language

We will use the enumerable set of variables and (infinite) set of constants of each type a. If n is a natural number and $a \in \Upsilon$, then $v_{n,a}$ is the n-th variable of type a, and Con_a is a set of constants of type a.

The language includes a set of meaningful expressions ME_a of each type a. It is defined recursively:

1. $v_{n,a} \in ME_a$; $Con_a \subseteq ME_a$.

2. If $\alpha \in ME_b$ and u is a variable of type a, $\lambda u\alpha \in ME_{\langle a,b\rangle}$.

3. If $\alpha \in ME_{\langle a,b\rangle}$ and $\beta \in ME_a$, then $\alpha(\beta) \in ME_b$.

4. If $\alpha, \beta \in ME_a$, then $\alpha = \beta \in ME_t$.

5. If $\varphi, \psi \in ME_t$ and u is a variable, then $\neg\varphi$, $[\varphi \wedge \psi]$, $[\varphi \vee \psi]$, $[\varphi \rightarrow \psi]$, $[\varphi \leftrightarrow \psi]$, $\forall u\psi$, $\exists u\psi$, $\Box\psi$, $W\psi$, and $H\psi \in ME_t$.

6. If $\alpha \in ME_a$, then $[^\wedge \alpha] \in ME_{\langle s,a \rangle}$.

7. If $\alpha \in ME_{\langle s,a \rangle}$, then $[^\vee \alpha] \in ME_a$.

8. There are no other meaningful expressions.

The language is the main technical tool to write formulas that are in the correspondence with CMs or their fragments. The set of formulas, however, is wider than the set of CMs.

3.2.3 Interpretation

Now let us introduce interpretation of intensional language. Let A and Asg be sets; A is a set of entities (or individuals), and Asg is a set of assignment points. Define the set $D_{a,A,Asg}$ of possible denotates of type a:

$$\begin{aligned} &D_{e,A,Asg} = A, \\ &D_{t,A,Asg} = \{0,1\}, \\ &D_{\langle a,b \rangle,A,Asg} = D_{b,A,Asg}{}^{D_{a,A,Asg}}, \\ &D_{\langle s,a \rangle,A,Asg} = D_{a,A,Asg}{}^{Asg}. \end{aligned} \tag{1}$$

As a rule, sets A and Asg are fixed. Under these conditions, we denote $D_{a,A,Asg} \equiv D_a$. We treat the interpretation as an ordered tuple:

$$\mathfrak{A} = \langle A, Asg, F \rangle, \tag{2}$$

where

1. A and Asg are non-empty sets.

2. F is a function whose domain is a set of constants.

3. If $a \in Y$ and $\alpha \in Con_a$, then $F(\alpha) \in D_{a,A,Asg}{}^{Asg}$.

We treat $\mathfrak{A}$-assignment as a function g, whose domain is a set of variables, such that when u is a variable of type a, then $g(u) \in D_{a,A,Asg}$. $G[x/u]$ means $\mathfrak{A}$-assignment:

$$g[x/u](v) = \begin{cases} x, & \text{if } u \equiv v, \\ g(v) & \text{otherwise.} \end{cases} \tag{3}$$

We define the intension $\alpha^{\mathfrak{A},g}$ and the extensional $\alpha^{\mathfrak{A},Asg,g}$ meaningful expression α when using the usual recursive definition. Complete form of definition is presented in [1]. The intension is the possible value of CM, and interpretation is a tool for the evaluation of CMs.

4. Problem domain description with CMs

The interpretation of the constructions, composing the CM, is made up in the framework of type theory with dependent types of functions and pairs. The interpretation of CM is considered as an object, the type of which can be constructed.

In the framework of the type system, a judgment can be expressed that an object has a particular type, or that two objects are equivalent. The equivalence of objects implies the equivalence of the corresponding CMs. The types of system objects are constructed in the form of an inductive class, and this allows deriving the properties of objects from the properties of their parts.

We consider two basic types of judgments:

$$\begin{gathered} a : A, \\ a = b : A. \end{gathered} \tag{4}$$

The judgment of the first form is interpreted as "a is an object (CM) of type A." The judgment of the second form is interpreted as "a and b are objects (CM's) of type A, equal by definition." Judgments can depend on assumptions of the form $x : A$. The collection of all such assumptions forms context for the judgment.

We use for the expression "A is a type" the formal notation $A : U$. Here U is a universe. Elements of the universe are types. We consider U as the type big enough to hold all the types that are necessary for the description of a given class of CMs, but we do not consider it as holding all possible types. Inaccurate definition of U can lead to paradoxes (e.g., if we consider $U : U$). It is possible to show that U can be defined in the type theory without paradoxes, but the proof of this fact is beyond the scope of the present work. U can be used for the representation of collection of types varying over a given type A.

In the type theory, we can construct new types from given ones. New types are equipped with functions for their construction and computation. The first construction is a function type. From the types A and B, we can construct the type $A \to B$ of functions with the domain A and codomain B. The construction rule for the elements of this type is called lambda abstraction. If we assume that $x : A$ and consider the expression $F : B$, then

$$(\lambda(x : A).F) : A \to B. \tag{5}$$

We can omit the type for the variable and write $\lambda x.F$. We adopt the usual convention that λ makes the variable bound and the variable which is not bound is free.

The computation rule for this type is

$$(\lambda x.F)a = F[x{:=}a], \tag{6}$$

where $F[x{:=}a]$ is a result of a substitution of all free occurrences of x to a. The pattern of CM for the functional type is show in **Figure 2a**; the process of currying is shown in **Figure 2b**.

The next construction is a dependent function type. The elements of such type are functions whose codomain may depend on the element of the domain to which the function is applied. From the type $A : U$ and family $B(x) : A \to U$, we can construct the type $\Pi_{x:A} B(x) : U$. If we have an expression $F : B(x)$ depending on $x : A$, then

$$(\lambda(x : A).F) : \Pi_{x:A}\, B(x). \tag{7}$$

We can apply a dependent function $f : \Pi_{x:A}\, B(x)$ to an argument $a : A$ to obtain an element $f(a) : B(a)$. In particular, we can define functions taking types as arguments. This possibility leads to the representation of polymorphic functions. The pattern of CM for dependent functional type is shown in **Figure 3a**.

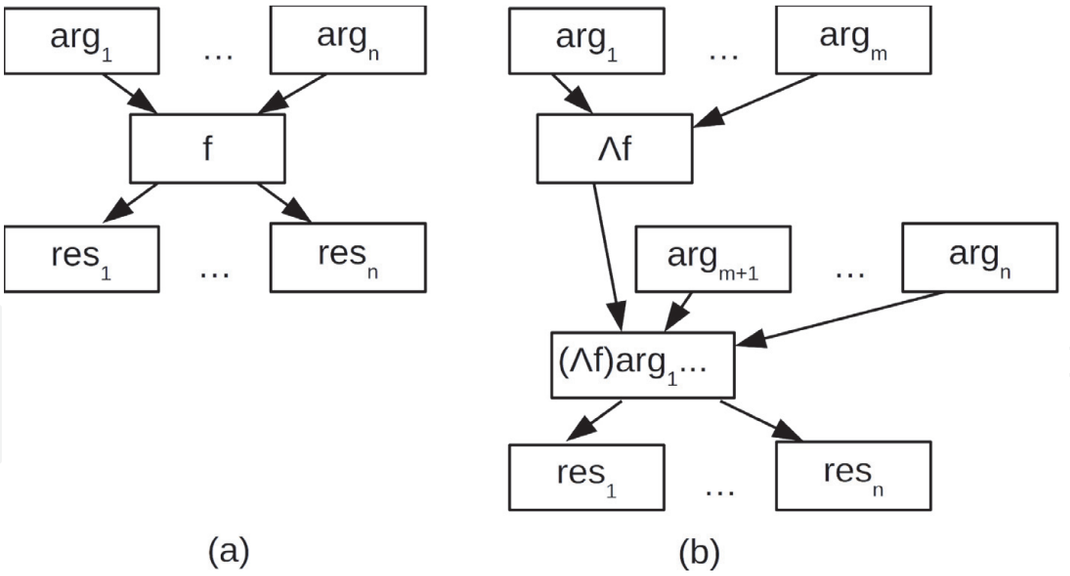

Figure 2.
The pattern of functional type: (a) multiargument function and (b) curried function.

The next construction is Cartesian product type. From the types A and B, we can construct the type $A \times B$. We also consider a product type with arity 0, which is called the unit type $1 : U$. For $a : A$ and $b : B$, we can construct $(a, b)A \times B$. The unit type has one element which we denote $* : 1$.

For the computation with product types, we have to define functions which have the elements of $A \times B$ as arguments (i.e., the function f of the type $A \times B \to C$). Hence we consider such elements as pairs, and we can make computations on such element with the function $g : A \to B \to C$ taking the components of the pair as arguments. Then we can define f as follows:

$$f(a, b) = gab. \tag{8}$$

We can consider the universal case and define the function

$$rec_{A\times B} : \Pi_{C:U}(A \to B \to C)A \times B \to C \tag{9}$$

with the equation

$$rec_{A\times B}Cg(a, b) = gab. \tag{10}$$

With this function, for example, we can define projections:

$$\begin{aligned} p &= rec_{A\times B}A(\lambda a.\lambda b.a) \\ q &= rec_{A\times B}B(\lambda a.\lambda b.b) \end{aligned} \tag{11}$$

Similarly for the unit type 1, we have

$$rec_1 : \Pi_{C:U}C \to 1 \to C \tag{12}$$

with the equation

$$rec_1 Cc* = c. \tag{13}$$

The pattern of CM for Cartesian product is shown in **Figure 3b**.

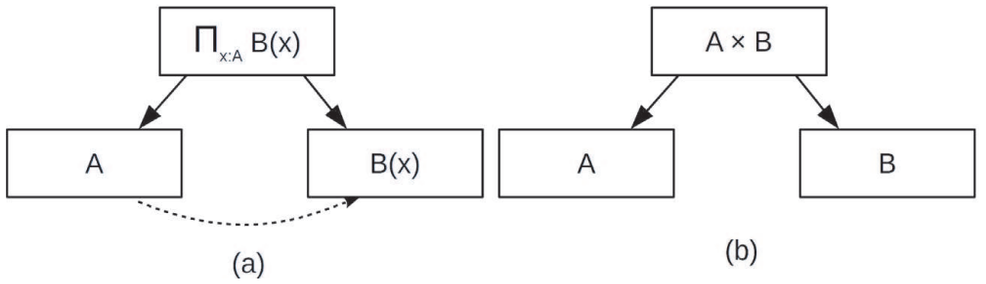

Figure 3.
The pattern of CM for (a) dependent functional type and (b) Cartesian product.

The Cartesian product represents pairs which components have independent types. So the next construction is dependent pair type. From the type $A : U$ and family $B : A \to U$, we can construct the type $\Sigma_{x:A}B(x) : U$. The construction on the dependent pair is generalization of the construction for product types. To define a function over a dependent pair type $f : \Sigma_{x:A}B(x) \to C$, we provide a function $g : \Pi_{x:A}B(x) \to C$ via the defining equation

$$f(a,b) = gab. \tag{14}$$

Similarly to the Cartesian product, we can define

$$rec_{\Sigma_{x:A}B(x)} : \Pi_{C:U}(\Pi_{x:A}B(x) \to C) \to (\Sigma_{x:A}B(x)) \to C \tag{15}$$

with the equation

$$rec_{\Sigma_{x:A}B(x)}\, Cg\,(a,b) = gab. \tag{16}$$

The pattern of CM for dependent pair type is shown in **Figure 4a**.

The last construction that we consider here is the sum type. From the types A and B, we can construct the type $A + B$. We also consider a sum type with arity 0, which is called the empty type $0 : U$. There are two ways to construct the elements of $A + B$, one is $inl(a) : A + B$ for $a : A$ and another is $inr(b) : A + B$ for $b : B$. The empty type has no elements.

To construct a function $f : A + B \to C$, we need functions $g : A \to C$ and $h : B \to C$. Then f is defined via the equations

$$\begin{aligned} f(inl(a)) &= ga, \\ f(inr(b)) &= hb. \end{aligned} \tag{17}$$

So we can consider the universal case

$$rec_{A+B} : \Pi_{C:U}(A \to C) \to (B \to C)A + B \to C \tag{18}$$

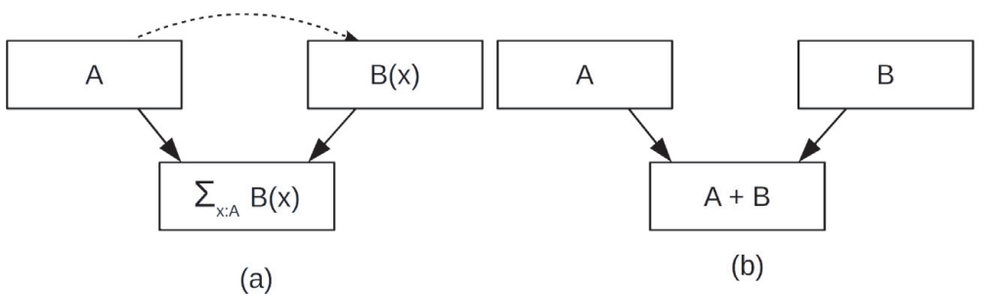

Figure 4.
The pattern of CM for (a) dependent functional type and (b) Cartesian product.

with the equations

$$\begin{aligned} rec_{A+B}\, C g h\,(inl(a)) &= g\,a, \\ rec_{A+B}\, C g h\,(inr(b)) &= h\,b. \end{aligned} \tag{19}$$

The pattern of CM for sum type is shown in **Figure 4b**.

5. Naming functors and their properties

Analysis of the proposed CM formalization suggests identifying the structure of CM models defined in the framework of type theory. The practice shows that an adequate approach in constructing such models is the category theory. At the same time, the model is built in the selected category using its objects and arrows. The objects are matched to the types used when constructing the CM, and arrows are matched to the nodes and links of the CM. Depending on the definition of a category, its objects can carry an additional structure within themselves. They can be put into line with the characteristics of the modeled domain.

Common constructions of the category theory can be used to introduce and process the CM constructions that are of interest from both a fundamental and a practical point of view. One of such constructions is the adjoint functor. It turns out that the conjugation scheme allows the transition from an "atomic" description of model objects, in which we distract from their internal structure to the description that takes into account such a structure. A back transition is also possible.

From a fundamental point of view, the mentioned feature provides the CM scaling, that is, folding the nested CM into a separate node and back deployment. From a practical point of view, this possibility corresponds to the definition of an abstract type of data within the framework of a programming system. In this case, the functors that implement conjugation provide a transition from the description of an abstract object or from the name of the object (which in the programming system corresponds to the address of the object in memory) to the representation of the object. This justifies their title as "naming" functors.

5.1 Adjunction

The traceability of the interpretation of various entries of an information object (CM or its fragment) requires constructing models of interpretation depending on the context. When the CM is placed in a programming environment (e.g., Java), the context is formed by the constructions chosen to represent the CM fragments, as well as the associated data. The context dependency can be traced basing on the use of the technique of variable domains, which are functors.

Practical implementation of the system supporting the work with CMs requires selecting a method of their representation by the programming structures, such as arrays or strings. A link to the filled-in fragment of the representing structure can be given as an index in the structure, which may be considered as a special naming structure. The work with CM necessitates a transition from the name of the CM fragment to its value and backward, i.e., naming and dereferencing the CM fragments. To ensure the completeness of the computational model, it must provide a transition from the naming constructs to the content of the representing constructions and back.

Because of this, it looks attractive to develop technology, (1) coordinating with the general structure of the computational model, i.e., presented in the form of a functor; (2) making possible to name structures and methods of working with

named structures; and (3) providing the ability to display on the structures of the programming system. Further on, it will be shown that in some cases the naming may be associated with the presenting CM's constructions in a categorically invariant manner. The corresponding categories contain structures of a certain type as objects, and the mappings between objects that preserve their structure serve as arrows. The construction of the required type model uses the concept of adjunction.

Definition. Adjunction between categories $\mathcal{C}$ and $\mathcal{D}$ is the four-tuple $\langle F, G, \varphi, \psi \rangle$, where F is a functor $\mathcal{C} \to \mathcal{D}$, G is a functor $\mathcal{D} \to \mathcal{C}$,

$$\mathcal{C} \underset{G}{\overset{F}{\rightleftarrows}} \mathcal{D}, \tag{20}$$

function φ (correspondingly ψ) to every pair of objects $c \in Ob(\mathcal{C})$, $d \in Ob(\mathcal{D})$ puts in correspondence the mapping $\mathcal{D}(Fc, d) \to \mathcal{C}(c, Gd)$ (correspondingly $\mathcal{C}(c, Gd) \to \mathcal{D}(Fc, d)$), natural according to c and d, and in this case $\varphi \circ \psi = 1$, $\psi \circ \varphi = 1$.

Remark. So, $\varphi_{c,d}$ and $\psi_{c,d}$ set the bijection

$$\mathcal{D}(Fc, d) \cong \mathcal{C}(c, Gd), \tag{21}$$

which is natural in c and d.

The given definition needs to be explained. Mind that the natural transformation from the functor $\mathcal{S} : \mathcal{C} \to \mathcal{D}$ into the functor $T : \mathcal{C} \to \mathcal{D}$ is the mapping μ, putting in correspondence with every object $a \in Ob(C)$ the arrow $\mu_a : Sa \to Ta \in Ar(D)$ in such a way that for each arrow $f : a \to b$ from $Ar(C)$, the diagram

$$\begin{array}{ccc} Sa & \overset{\mu_a}{\to} & Ta \\ \downarrow Sf & & \downarrow Tf \\ Sb & \overset{\mu_b}{\to} & Tb \end{array} \tag{22}$$

commutates, i.e., $Tf \circ \mu_a = \mu_b \circ Sf$.

Also mind that with the help of $\mathcal{C}(a, b)$, the set of arrows is designated as the arrows of category $\mathcal{C}$ with start in the object a and the end in the object b (under the "set" we understand the set in that model of set theory, in which the category $\mathcal{C}$ is defined). In such a way, $\mathcal{C}(-, -)$ is the mapping of the pair of objects of the category $\mathcal{C}$ into the sets. This mapping is an object mapping of *bifunctor* (i.e., of a functor from two parameters), contravariant in the first argument and covariant in the second one. In full this bifunctor is defined as follows:

$$\begin{aligned} &\mathcal{C}(a, b) = \{ f \in Ar(\mathcal{C}) \,|\, f : a \to b \}, \\ &\mathcal{C}(h, k) = k \circ f \circ h. \end{aligned} \tag{23}$$

In accordance with the following diagram

$$a' \overset{h}{\to} a \overset{f}{\to} b \overset{k}{\to} b', \tag{24}$$

where $h : a' \to a$, $k : b \to b'$.

To give a formalized representation of this functor, it is convenient to pass over to the dual category $\mathcal{C}^{op}$. Then the functor defined above turns out to be a functor from $\mathcal{C}^{op} \times \mathcal{C} \to \mathcal{S}et$.

Now let us consider the constructions $\mathcal{D}(Fc,d)$ and $\mathcal{C}(c,Gd)$ from the definition of adjunction. Absolutely similar to the previous, they may be considered as object mappings of bifunctors $\mathcal{D}(F-,-)$ and $\mathcal{C}(-,G-)$, acting from $\mathcal{C}^{op}\times\mathcal{D}$ in *Set* in accordance with the following rule:

$$\begin{aligned}&\mathcal{D}(F-,-):\mathcal{C}^{op}\times\mathcal{D}\overset{F^{op}\times Id}{\to}\mathcal{D}^{op}\times\mathcal{D}\to Set,\\&\mathcal{C}(-,G-):\mathcal{C}^{op}\times\mathcal{D}\overset{Id\times G}{\to}\mathcal{C}^{op}\times\mathcal{C}\to Set.\end{aligned}\tag{25}$$

Totally these functors are set by the following way:

$$\begin{aligned}\mathcal{D}(Fa,b)&=\{f\in Ar(\mathcal{D})\,|f:Fa\to b\,\}\\\mathcal{D}(Fh,k)&=k\circ f\circ Fh,\end{aligned}\tag{26}$$

with the diagram (containing the arrows from $Ar(\mathcal{D})$)

$$Fa'\overset{Fh}{\to}Fa\overset{f}{\to}b\overset{k}{\to}b',\tag{27}$$

and

$$\begin{aligned}\mathcal{C}(a,Gb)&=\{f\in Ar(\mathcal{C})|f:a\to Gb\}\\\mathcal{C}(h,Gk)&=Gk\circ f\circ h,\end{aligned}\tag{28}$$

with the diagram (in $Ar(\mathcal{C})$)

$$a'\overset{h}{\to}a\overset{f}{\to}Gb\overset{Gk}{\to}Gb',\tag{29}$$

where $h:a'\to a\in Ar(\mathcal{C})$ and $k:b\to b'\in Ar(\mathcal{D})$.

It is indicated in the definition of adjunction that the mappings φ and ψ are natural transformations of the above defined functors (i.e., arrow in the category $Funct(\mathcal{C}^{op}\times\mathcal{D},Set)$), and this finetunes the phrase "φ and ψ are natural for c and d." Let us consider the diagrams that describe this naturality:

$$\begin{array}{ccc}\mathcal{D}(Fc,d) & \overset{\varphi_{c,d}}{\to} & \mathcal{C}(c,Gd)\\ \downarrow\mathcal{D}(Fh,k) & & \downarrow\mathcal{C}(h,Gk)\\ \mathcal{D}\left(Fc',d'\right) & \overset{\varphi_{c',d'}}{\to} & \mathcal{C}\left(c',Gd'\right)\end{array}\tag{30}$$

The arrow in the category $\mathcal{C}^{op}\times\mathcal{D}$ is the pair of arrows $\langle h,k\rangle$, where $h:c'\to c\in Ar(\mathcal{C})$ and $k:d\to d'\in Ar(\mathcal{D})$. We have

$$\mathcal{C}(h,Gk)\circ\varphi_{c,d}=\varphi_{c',d'}\circ\mathcal{D}(Fh,k).\tag{31}$$

The values of the considered functors are ordinary sets; this is why the above written correlation binds ordinary functions on sets. Thanks to this it is possible to compare the values of functions on an arbitrary element of their definition, which is the function $f\in\mathcal{D}(Fc,d)$ (i.e., $f:Fc\to d\in Ar(\mathcal{D})$), i.e., to apply both parts of the equality to this function. We get

$$\begin{aligned}\left(\mathcal{C}(h,Gk)\circ\varphi_{c,d}\right)f&=\left(\varphi_{c',d'}\circ\mathcal{D}(Fh,k)\right)f\\\mathcal{C}(h,Gk)\left(\varphi_{c,d}f\right)&=\varphi_{c',d'}\left(\mathcal{D}(Fh,k)f\right)\\Gk\circ\varphi_{c,d}f\circ h&=\varphi_{c',d'}(k\circ f\circ Fh)\end{aligned}\quad.\tag{32}$$

Schematically the action φ may be shown as follows:

$$\begin{array}{lccccccc} Ar(\mathcal{D}): & \varphi_{c',d'}(& Fc' & \overset{Fh}{\to} & Fc & \overset{f}{\to} & d & \overset{k}{\to} & d' &) \\ Ar(\mathcal{C}): & & c' & \overset{h}{\to} & c & \overset{\varphi_{c,d}f}{\to} & Gd & \overset{Gk}{\to} & Gd'. & \end{array} \tag{33}$$

The drawn ratio is equivalent to the combination of two of its private cases:

$$Gk \circ \varphi_{c,d} f = \varphi_{c',d'}(k \circ f), \qquad \varphi_{c,d} f \circ h = \varphi_{c',d'}(f \circ Fh), \tag{34}$$

and to work with which is somehow easier than with the initial ration as each of them contains only one additional arrow.

Absolutely similar the condition of naturality for ψ is expressed as follows:

$$\begin{array}{ccc} \mathcal{C}(c, Gd) & \overset{\psi_{c,d}}{\to} & \mathcal{D}(Fc, d) \\ \downarrow \mathcal{C}(h, Gk) & & \downarrow \mathcal{D}(Fh, k) \\ \mathcal{C}(c', Gd') & \overset{\psi_{c',d'}}{\to} & \mathcal{D}(Fc', d') \end{array} \tag{35}$$

or

$$\mathcal{D}(Fh, k) \circ \psi_{c,d} = \psi_{c',d'} \circ \mathcal{C}(h, Gk). \tag{36}$$

Selecting the arrow $g \in \mathcal{C}(c, Gd)$ (i.e., $f : c \to Gd \in Ar(\mathcal{C})$), we get

$$\begin{array}{c} k \circ \psi_{c,d} g \circ Fh = \psi_{c',d'}(Gk \circ g \circ h), \\ k \circ \psi_{c,d} g = \psi_{c',d'}(Gk \circ g), \qquad \psi_{c,d} g \circ Fh = \psi_{c',d'}(f \circ h), \end{array} \tag{37}$$

schematically

$$\begin{array}{lccccccc} Ar(\mathcal{C}): & \psi_{c',d'}(& c' & \overset{h}{\to} & c & \overset{g}{\to} & Gd & \overset{Gk}{\to} & Gd' &) \\ Ar(\mathcal{D}): & & Fc' & \overset{Fh}{\to} & Fc & \overset{\psi_{c,d}g}{\to} & d & \overset{k}{\to} & d'. & \end{array} \tag{38}$$

In case of adjunction, the functor F is named *left adjoint* to the functor G, and G is *right adjoint* to the functor F. To make the remembering easier, note that in the bijection $\mathcal{D}(Fc, d) \cong \mathcal{C}(c, Gd)$, the functor F is applied to the argument in the left position and G is applied to the argument in the right position.

In the given definition the category, *Set* plays a special role as a category, in which the sets of arrows are defined for $\mathcal{C}$ and $\mathcal{D}$. This restriction can be overcome: to define a category, another category can be used as a basis, and this category should be additionally conditioned.

5.2 Samples of naming constructions

Let us consider some examples of naming constructions based on adjoint functor.

5.2.1 Arrays (vectors)

One of the standard representing constructions in practical programming languages for complex structures of data, including CMs or their fragments, is an array or vector. The bunch of possible vectors forms a linear space.

Let us consider a common structure of linear space. Let K be a field and $\mathcal{V}ect_K$ be the category of linear (vector) spaces over the field K, whose objects are vector

spaces and arrows are linear mappings between them. We will consider not only finite-dimensional but also infinite-dimensional spaces. This practically means that we do not limit the dimension of the represented CM. In this situation we assume that the elements of the infinite-dimensional linear space are finite linear combinations of the vectors of the space with coefficients from the field K (in accordance with the fact that we do not specify on the space any topology and, thuswise, convergence). In this scenario, obviously, the axioms of the linear vector space turn out to be fulfilled.

Let us consider the functor $G : \mathcal{V}ect_K \to Set$, which associates with each vector space of its underlying set, i.e., the set of vectors of this space, considered as nonfactorable (structureless) objects. Obviously, G is a functor; it is the so-called neglecting, erasing, or forgetful functor, which is so named because it forgets the structure of objects of the initial category or its part.

To make it clearer, let us consider the vector space B and suggest that the vectors b_i form a basis in it. Then each vector of the space B can be expressed as $v = \sum_{b_i \in B' \subset B} k_i b_i$, where $k \in K$ and B' are finite. As a result of the action of the functor G , the space B is transformed into the set GB consisting of the same elements as the underlying set of B. The vector v, considered as an element of GB, will be written in the form $\ulcorner v \urcorner$, where the angle brackets $\ulcorner \ldots \urcorner$ show that we consider the object as having no internal structure (i.e., and actually form the name of the corresponding object or, more precisely, an expression describing this object). The arrows $g \in Ar(\mathcal{V}ect_K)$ are transferred to "the same" arrows, considered on the sets, i.e.,

$$Gg(\ulcorner v \urcorner) = \ulcorner g(v) \urcorner. \tag{39}$$

The considered functor has the left adjoint $F : \mathcal{S}et \to \mathcal{V}ect_K$. The functor F for every set A forms vector space, in which the elements of the initial set $a \in A$ are the basis and are considered as their formal finite linear combinations $\sum_{a_i \in A'} k_i a_i$, where $k_i \in K$ and $A' \subset A$ are finite. The function $f : A \to B$ is mapped by the functor F into linear operator, which acts on the basic vectors of the space FA (i.e., on the elements of the set A) in the same way as the function f (i.e., transforms $a \in FA$ to $f(a) \in FB$), and on linear combinations $a \in A' \subset A$ it continues formally according to the linearity:

$$Ff\left(\sum_{a_i \in A'} k_i a_i\right) = \sum_{a_i \in A'} k_i f(a_i). \tag{40}$$

It is easy to check that the given definition really specifies the functor.

To establish the adjunction it is necessary to specify φ and ψ and after it to check their naturalness. Since $\varphi : \mathcal{V}ect_K(FA, B) \to \mathcal{S}et(A, GB)$, we need to consider the arrow $f : FA \to B$ and put into correspondence with it the arrow $\varphi f : A \to GB$. It's easy to do due to FA containing all linear combinations of the type $\sum_{a_i \in A' \subset A} k_i a_i$, and it also contains, in particular, such linear combinations, where $A' = \{a_i\}$ and the corresponding $k_i = 1$. Then we can write

$$\sum_{a_i \in \{a_i\}} 1 \cdot a_i = 1 \cdot a_i = a_i. \tag{41}$$

Such elements can be identified with elements of the set A. Since f is given on all elements of the set FA, it is also set on the elements identified with elements of the set A, and it puts into correspondence with them some elements of the set B. But φf must be specified on the elements of the set A and associate them with some elements of

the set GB. It is easy to see what these elements are: they must correspond to the initial elements of B under the correspondence given by the functor G. Formally

$$(\varphi f)(a) = \ulcorner f(a) \urcorner, \tag{42}$$

where $a \in A$.

Let us check the naturalness of φ. For this purpose we consider the arrows $h : A' \to A$ and $k : B \to B'$, where $A, A' \in Ob(Set)$ and $B, B' \in Ob(Vect_K)$. We have $f' = k \circ f \circ Fh : FA' \to B'$, this is why $\varphi f' : A' \to GB'$. Two arrows are equal in the category Set if their values are equal on every element of their domain. Let us consider the element $s \in A'$ and compute on it the values of the arrows forming the naturality diagram:

$$\begin{aligned} &\varphi(k \circ f \circ Fh)(s) = \\ &= \ulcorner (k \circ f \circ Fh)(s) \urcorner = \\ &= \ulcorner k(f(Fh(s))) \urcorner = \\ &= \ulcorner k(f(h(s))) \urcorner \end{aligned} \tag{43}$$

and

$$\begin{aligned} &(Gk \circ \varphi f \circ h)(s) = \\ &= Gk((\varphi f)(h(s))) = \\ &= Gk(\ulcorner f(h(s)) \urcorner) = \\ &= \ulcorner k(f(h(s))) \urcorner. \end{aligned} \tag{44}$$

Since the values coincide for an arbitrary s, the corresponding arrows also coincide. Thus, the naturalness of φ is proven.

Now let us define $\psi : \mathcal{S}et(A, GB) \to \mathcal{V}ect_K(FA, B)$. We'll consider the arrow $g : A \to GB$. The value of this function for any $a \in A$ can be represented as $\ulcorner b \urcorner$ for some $b \in B$. Since the mapping $\lambda x.\ulcorner x \urcorner$ is one-to-one, there is a single function that can be associated with the function g; this is the function $g^* : A \to B$, for which

$$g(a) = \ulcorner g^*(a) \urcorner. \tag{45}$$

The function g^* in some specified sense "makes the analysis" of the element a, giving the permit to pass over from "structureless object" $\ulcorner b \urcorner$ (the element of the set GB) to the "structured object" b (the element of the vector space B).

We need the behavior g^* related to the composition. Let $h : A' \to A$. Then according to the definition, $(g \circ h)(a) = \ulcorner (g \circ h)^*(a) \urcorner$. Furthermore, $(g \circ h)(a) = g(h(a)) = \ulcorner g^*(h(a)) \urcorner$. Consequently,

$$(g \circ h)^* = g^* \circ h. \tag{46}$$

The situation is cleared somehow by the diagram

$$\begin{array}{ccccc} A' & \stackrel{h}{\to} & A & \stackrel{g}{\to} & GB \\ \downarrow (g \circ h)^* & & \downarrow g^* & & \uparrow \ulcorner \urcorner \\ B & = & B & = & B, \end{array} \tag{47}$$

where $\ulcorner \urcorner$ is a function $\lambda x.\ulcorner x \urcorner : B \to GB$.

Now let $k : B \to B'$, then $Gk : GB \to GB'$. According to the definition, $(Gk \circ g)(a) = \ulcorner (Gk \circ g)^*(a) \urcorner$. But $(Gk \circ g)(a) = Gk(g(a)) = Gk(\ulcorner g^*(a) \urcorner) = \ulcorner k(g^*(a)) \urcorner$. Consequently,

$$(Gk \circ g)^* = k \circ g^*, \tag{48}$$

and this may be illustrated by the diagram

$$\begin{array}{ccccc} A & \overset{g}{\to} & GB & \overset{Gk}{\to} & GB' \\ \| & & \uparrow\ulcorner\urcorner & & \uparrow\ulcorner\urcorner \\ A & \overset{g^*}{\to} & B & \overset{k}{\to} & B'. \end{array} \tag{49}$$

Now we are ready to define the arrow $\psi g : FA \to B$ with the values in the vector space B. It may be determined in such a way

$$(\psi g)\left(\sum_{a_i \in A'} k_i a_i\right) = \sum_{a_i \in A'} (k_i g^*(a_i)). \tag{50}$$

Let us check the naturalness of the ψ. Once again we'll consider the arrows $h : A' \to A$ and $k : B \to B'$, where $A, A' \in Ob(\mathcal{S}et)$ and $B, B' \in Ob(\mathcal{V}ect_K)$. We have $g' = Gk \circ g \circ h : A' \to GB'$; this is why $\psi g' : FA' \to B'$. Let us consider the element $t = \sum_{a_i \in A'} k_i a_i \in FA'$ and compute on it the value of the arrows that form the naturality diagram:

$$\begin{aligned} &\psi(Gk \circ g \circ h)(t) = \\ &= \psi(Gk \circ g \circ h)\left(\sum_{a_i \in A'} k_i a_i\right) = \\ &= \sum_{a_i \in A'} k_i \cdot (Gk \circ g \circ h)^*(a_i) = \\ &= \sum_{a_i \in A'} k_i \cdot k(g^*(h(a_i))) \end{aligned} \tag{51}$$

and

$$\begin{aligned} &(k \circ \psi g \circ Fh)(t) = \\ &= (k \circ \psi g \circ Fh)\left(\sum_{a_i \in A'} k_i a_i\right) = \\ &= k\left(\psi g\left(Fh\left(\sum_{a_i \in A'} k_i a_i\right)\right)\right) = \\ &= k\left(\psi g\left(\sum_{a_i \in A'} k_i \cdot h(a_i)\right)\right) = \\ &= k\left(\sum_{a_i \in A'} k_i \cdot g^*(h(a_i))\right) = \\ &= \sum_{a_i \in A'} k_i \cdot k(g^*(h(a_i))). \end{aligned} \tag{52}$$

The values coincide for arbitrary t, and this proves the naturality of ψ.
This example is important because of three reasons:

1. The construction of the vector space is one of the oldest constructions for data representation and at the same time one of the simplest well-known algebraic constructions, and it is an example of a typical algebraic system (underlying set and a signature given by a set of axioms); therefore the construction of an adjoint functor demonstrates some typical common features of such functors.

2. The category of vector spaces, as distinguished, for example, from the category of monoids, hardly uses mechanisms specific to abstract data types (such as a queue or a stack), except for the proper neglecting functor inducing nesting of named objects of the corresponding categories; thanks to this the construction of the attachment is clearly seen in this example.

3. The vector spaces are well studied; this allows without a long introduction to use constructions of considerable generality, which must be specially constructed for other categories.

The above shown example, in particular, demonstrates that the adjunction does not formalize in any sense the idea of mutually inverse functors. We see that for a finite-dimensional space B over an infinite field K, the set GB is infinite, and, consequently, the space FGB is infinite-dimensional, i. e., it is not isomorphic to the initial one. On the contrary, the adjunction acts as a mechanism for assigning a given structure with an additional structure (in our example—any set with a vector space structure) and operations on this additional structure, connecting it with the original one.

Taking into account the foregoing, the above given example demonstrates that conjunctions can be used to describe abstract data types (in our example, to describe an abstract vector). In the example, the mapping φ demonstrates encapsulation, i.e., hiding the internal structure of the object, and ψ provides the definition of operations on the encapsulated type of data.

In general case the functor specifies the correspondence of objects of one category with objects of another and does not suggest the presence of any arrows between these objects. However, in the given example, the fact that the vector space is defined as a set with an additional structure makes it possible to consider the function g^* on the set with values in a vector space, as well as the inverse function $\lambda x.\ulcorner x \urcorner$. This construction is not valid in the general case.

5.2.2 Monoids

Let us consider another structure used to represent CMs—strings. The strings can be used to name CM's elements or bind additional textual information to CM. It appears that it is possible to formalize naming procedures for strings based on the category of monoids. Let us show this.

Let $\mathscr{M}on$ be the category of monoids. Mind that a monoid is a triple $M = \langle M, \cdot, e \rangle$, where M is the underlying set, the dot $(\cdot)$ is the binary operation on the set M, and $e \in M$ is the identity element of the monoid. Then the objects of category $\mathscr{M}on$ are the monoids, and the arrows are the mappings keeping the monoid structure.

Let us consider the neglecting functor $G : \mathscr{M}on \to \mathscr{S}et$, which associates each underlying set with its monoid and to the mapping of the monoids to the corresponding mapping of the sets. Similar to the previous example, we'll mark $\ulcorner m \urcorner$

element GM, corresponding to the element m of the initial monoid M. Like in the previous example, we have

$$Gg(\ulcorner m \urcorner) = \ulcorner g(m) \urcorner \tag{53}$$

for the arrow g between the monoids.

This functor has left adjoint $F : \mathcal{S}et \to \mathcal{M}on$, which is defined as follows. For the set A, the underlying set of the monoid FA is the set of finite sequences $\langle a_1, \ldots, a_n \rangle$ of the elements $a_i \in A$, including the empty sequence $\langle\rangle$. The monoid operation is defined by

$$\langle a_1, \ldots, a_n \rangle \cdot \langle a_{n+1}, \ldots, a_{n+m} \rangle = \langle a_1, \ldots, a_n, a_{n+1}, \ldots, a_{n+m} \rangle, \tag{54}$$

and the identity element of the monoid FA is $\langle\rangle$. It is easily checked that the indicated construction correctly specifies the monoid.

For the function $f : A \to B$, the action of the functor F is specified as follows:

$$Ff(\langle a_1, \ldots, a_n \rangle) = \langle f(a_1), \ldots, f(a_n) \rangle. \tag{55}$$

It is easy to note that F is really a functor.

For testing the adjunction, it needs to specify φ and ψ and verify that the conditions in the definition of conjoint functors are fulfilled. To set $\varphi : \mathcal{M}on(FA, M) \to \mathcal{S}et(A, GM)$, let us consider the arrow $f : FA \to M$ and put in correspondence with it the arrow $\varphi f : A \to GM$. Now we consider the values of the arrow f on FA elements of the form $\langle a \rangle$, i.e., single-element sequences. We put

$$(\varphi f)(a) = \ulcorner f(\langle a \rangle) \urcorner. \tag{56}$$

Let us check the naturality of the transformation defined by this way. We consider the arrows $h : A' \to A$ and $k : M \to M'$, where $A, A' \in Ob(\mathcal{S}et)$ and $M, M' \in Ob(\mathcal{M}on)$. We have $f' = k \circ f \circ Fh : FA' \to M'$; this is why $\varphi f' : A' \to GM'$. Let us consider the element $s \in A'$ and compute on it the values of the arrows that form the naturality diagram:

$$\begin{aligned} &\varphi(k \circ f \circ Fh)(s) = \\ &= \ulcorner (k \circ f \circ Fh)(\langle s \rangle) \urcorner = \\ &= \ulcorner k(f(Fh(\langle s \rangle))) \urcorner = \\ &= \ulcorner k(f(\langle h(s) \rangle)) \urcorner \end{aligned} \tag{57}$$

and

$$\begin{aligned} &(Gk \circ \varphi f \circ h)(s) = \\ &= Gk((\varphi f)(h(s))) = \\ &= Gk(\ulcorner f(\langle h(s) \rangle) \urcorner) = \\ &= \ulcorner k(f(\langle h(s) \rangle)) \urcorner. \end{aligned} \tag{58}$$

Now we'll specify $\psi : \mathcal{S}et(A, GM) \to \mathcal{M}on(FA, M)$. Let us consider the arrow $g : A \to GM$. Similar to the previous example with the function g, it is possible to associate the single function $g^* : A \to M$, for which

$$g(a) = \ulcorner g^*(a) \urcorner. \tag{59}$$

Likewise the previous example, it is possible to establish the properties g^* related to the compositions

$$(g \circ h)^* = g^* \circ h \tag{60}$$

and

$$(Gk \circ g)^* = k \circ g^*, \tag{61}$$

where $h : A' \to A$ and $k : M \to M'$. Now

$$(\psi g)\langle a_1, \ldots, a_n\rangle = g^*(a_1) \cdot \ldots \cdot g^*(a_n) = \prod_i g^*(a_i). \tag{62}$$

Let us check the naturality of ψ. Once again we consider the arrows $h : A' \to A$ and $k : M \to M'$, where $A, A' \in Ob(\mathcal{S}et)$ and $B, B' \in Ob(\mathcal{M}on)$. We have $g' = Gk \circ g \circ h : A' \to GM'$; this is why $\psi g' : FA' \to M'$. Let us consider the element $t = \langle a_1, \ldots, a_n\rangle \in FA'$ and compute on it the values of the arrows that form the naturality diagram:

$$\begin{aligned}
&\psi(Gk \circ g \circ h)(t) = \\
&= \psi(Gk \circ g \circ h)(\langle a_1, \ldots, a_n\rangle) = \\
&= \prod_i (Gk \circ g \circ h)^*(a_i) = \\
&= \prod_i (k \circ g^* \circ h)(a_i) = \\
&= \prod_i k(g^*(h(a_i)))
\end{aligned} \tag{63}$$

and

$$\begin{aligned}
&(k \circ \psi g \circ Fh)(t) = \\
&= (k \circ \psi g \circ Fh)(\langle a_1, \ldots, a_n\rangle) = \\
&= k(\psi g(Fh(\langle a_1, \ldots, a_n\rangle))) = \\
&= k(\psi g(\langle h(a_1), \ldots, h(a_n)\rangle)) = \\
&= k\left(\prod_i g^*(h(a_i))\right) = \\
&= \prod_i k(g^*(h(a_i))).
\end{aligned} \tag{64}$$

The naturality of ψ is proven by the results coincidence.

Comparing this example with the previous one, it is seen that the first example is indeed formally simpler, since it does not require the construction of sequences of elements. The sequences in this case can be considered as representing constructions that ensure the mapping of *Mon* to *Set*, meeting the requirements formulated at the beginning of the item. Indeed, (1) the formed constructions have a functorial nature, (2) the functions that provide naming and dereferencing are explicitly constructed, and (3) due to choosing the basic categories ($\mathcal{V}ect$ or $\mathcal{M}on$, respectively), the mapping of structures are ensured in the construction of the supporting programming environment. In this case the adjoint functors can be considered as a variant of the CM's representation technique by means of a practical programming system.

6. Conclusions

The chapter considered a variant of solving the problem to store the data in a web environment and provide an access to the data based on their semantics. The semantics of data may be referred both to ensuring that the information complies with the put restrictions and to the traceability of nature of the problems that are solved by the users of different classes. The data is assumed to describe a certain subject area.

To represent the semantic nature of the data in the work, a representation in the form of a semantic network was used. The semantic network was considered as a set of marked nodes and marked links connecting them. The chapter considered the ways to access the nodes of the network, providing both the omission of irrelevant nodes and the decomposition of nodes.

The tools of describing users and their means of access to data that takes into account the specifics of the tasks to be solved must combine enough power to distinguish the relevant elements of the description and simplicity. It makes it possible to practically use the descriptions without excessive detailing, traditionally leading to an increase in the volume and complexity of the description. The work used cognitive maps to describe subjective views on the domain.

The chapter determines the CMs as hierarchically organized sets of nodes connected by unlabeled links. CMs can also contain links between nodes that are not in a hierarchical relationship. Due to:

1. The CM's syntax which differs from the semantic network syntax

2. The possibility of late fixation of typed restrictions

3. The possibility that links are fixed in the later stages of map development and are not placed or fixed by error

The cognitive maps cannot be considered as semantic networks. However, it is possible to propose matching procedures that will make it possible to consider CMs as a special type of semantic networks.

To determine the language of the description of the subjects and subjective points of view on the data, the work used a variant of intensional logic language. The essential feature of the language is the possibility to construct expressions that are indexable during interpretation, which makes it possible to study and use the dependence of expressions on a parameter. A number of methods for constructing CMs are distinguished, each of which is associated with a formula of intensional logic.

The semantics of intensional logic is constructed basing on recursively defined intensions. The inclusion of lambda expressions in the language and the definition of the corresponding semantic construction provide the computational nature of semantics. The interpretation of quantifiers and operators as special types of applications (applications of functions to the argument) makes it possible to determine all constructions of the model as applicative ones and attributes a computational nature to the models.

The constructed semantics makes it possible to express constructions in the form of CMs; these constructions describe the subject area from the point of view of experts. The chapter shows the possibility of such an expression with the example of the homotopy theory of types. Ever basic construction of the theory of types is accompanied by its presentation in the form of a cognitive map. The use of dependent type theory provides a subjective description of the subject area.

Computational methods for representing CM's semantics can be promoted to the level of support for processing CMs by means of a programming system. The work develops a functor technique for this. The model constructions, naming semantic elements (CM's or their fragments), are mapped onto the constructions of the representing environment with the help of the technique of conjoint functors. In this way, the computational model can be extended to CM's support techniques. This approach ensures the correctness of the tool kits and reduces the time for their development.

In the whole the constructed computational model makes the basis for the description of subjective views on the subject area, their representation in the model, and placing in a supporting programming environment. Thus, the model can serve as the basis for the development technique and maintenance of tool kits to support the description of the domain based on CMs. The elements of the model were tested when developing the practical information systems in the field of legal regulation of the best available technology implementation.

Acknowledgements

The chapter is supported by the grants 19-07-00326-a, 19-07-00420-a, 18-07-01082-a, and 17-07-00893-a of the Russian Foundation for Basic Research.

Author details

Larisa Yu. Ismailova[1], Sergey V. Kosikov[2*] and Viacheslav E. Wolfengagen[1]

1 National Research Nuclear Institute "MEPhI", Moscow, Russia

2 Institute for Contemporary Education "JurInfoR-MGU", Moscow, Russia

*Address all correspondence to: kosikov.s.v@gmail.com

References

[1] Ismailova LY, Wolfengagen VE, Kosikov SV. A computational model for supporting access policies to semantic web. In: Proc. of the Ninth Annual Meeting of the BICA Society; 22–24 August 2018; Prague, Czech Republic. 2019. pp. 145-154

[2] Kosikov SV, Ismailova LY, Wolfengagen VE. Network modeling environment for supporting families of displaced concepts. In: Proc. of the Ninth Annual Meeting of the BICA Society; 22–24 August 2018; Prague, Czech Republic. 2019. pp. 187-196

[3] Wolfengagen VE, Ismailova LY, Kosikov SV. A computational model for refining data domains in the property reconciliation. In: 2016 Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications, Moscow, 6–8 July 2016; Moscow, Russia. 2016. pp. 58-63

[4] Kosikov SV, Ismailova LY, Wolfengagen VE. The presentation of evolutionary concepts. Advances in Intelligent Systems and Computing. 2018;**636**:113-125

[5] Ismailova LY, Kosikov SV, Wolfengagen VE. Data domains modeling situationally determined evaluation. In: Proc. of Intelligent Systems Conference (IntelliSys); 6–7 September 2018; London, UK. 2018. pp. 972-976

[6] Ismailova LY, Wolfengagen VE, Kosikov SV. Basic constructions of the computational model of support for access operations to the semantic network. In: Proc. of the 8th International Conference on Biologically Inspired Cognitive Architectures (BICA 2017); 1–6 August 2017; Moscow, Russia. 2018. pp. 83-188

[7] FreeMind. Available from: http://freemind.sourceforge.net/ [Accessed: 01 June 2019]

[8] MindMeister. Online Mind Mapping. Mind Mapping Software. Available from: https://www.mindmeister.com/ [Accessed: 01 June 2019]

[9] MindManager Mind Mapping Software. Available from: https://www.mindjet.com/mindmanager/ [Accessed: 01 June 2019]

[10] Cacoo. Mind Maps Guide. Available from: https://cacoo.com/resources/mind-maps-guide/ [Accessed: 01 June 2019]

[11] Mindmup. Tutorials and Guides. Available from: https://www.mindmup.com/tutorials/index.html [Accessed: 01 June 2019]

[12] XMind–Mind Mapping Software. Available from: https://www.xmind.net/developer/ [Accessed: 01 June 2019]

[13] Laboratory of Cognitive Modelling and Management of Development of Situations. Institute of Control Sciences of Russian Academy of Sciences. Available from: https://www.ipu.ru/node/11921 [Accessed: 17 May 2019]

[14] Bell DE, Raiffa H, Tversky A, editors. Decision Making: Descriptive, Normative and Prescriptive Interactions. Cambridge: Cambridge University Press; 1988. pp. 9-32

[15] Chrastil ER, Warren WH. From cognitive maps to cognitive graphs. PLoS One. 2014;**9**:e112544. DOI: 10.1371/journal.pone.0112544

[16] Ferreira Opaso EV, Terelanskiy PV. Representation of cognitive maps in 3-dimensional space. In: VSPU-2014, Moscow. 2014

[17] Kotseruba I, Tsotsos JK. 40 years of cognitive architectures: Core cognitive abilities and practical applications. Artificial Intelligence Review. 2018. Available from: https://doi.org/10.1007/s10462-018-9646 [Accessed: 01 June 2019]

[18] Jung JJ, Jung KY, Jo GS. Ontological cognitive map for sharing knowledge between heterogeneous businesses. In: Yazıcı A, Şener C, editors. Computer and Information Sciences. ISCIS. 2003. p. 2869

[19] Furnari S. A cognitive mapping approach to business models: Representing causal structures and mechanisms. Advances in Strategic Management. 2015;**33**:207-239. DOI: 10.1108/S0742-332220150000033025

[20] Kluss T, Marsh WE, Zetzsche C, et al. Representation of impossible worlds in the cognitive map. Cognitive Processing. 2015;**16**:271. DOI: 10.1007/s10339-015-0705-x

[21] Chaib-draa B. Cognitive maps: Theory, implementation, and practical applications in multiagent environments. IEEE Transactions on Knowledge and Data Engineering. 2002; **14**(6):1201-1217

[22] Tarski A. On the calculus of relations. Journal of Symbolic Logic. 1941;**6**(3):73-89

[23] Miao YA. Cognitive map network model. In: Bai Q, Fukuta N, editors. Advances in Practical Multi-Agent Systems. Studies in Computational Intelligence. Vol. 325. Berlin, Heidelberg: Springer; 2002

[24] Concepción L, Nápoles G, Grau I, Vanhoof K, Bello R. Retracted chapter: Towards the convergence in fuzzy cognitive maps based decision-making models. In: Berger-Vachon C, Gil Lafuente A, Kacprzyk J, Kondratenko Y, Merigó J, Morabito C, editors. Complex Systems: Solutions and Challenges in Economics, Management and Engineering. Studies in Systems, Decision and Control. Vol. 125. Cham: Springer; 2018

[25] Kireev VS. Development of fuzzy cognitive map for optimizing e-learning course. In: Kalinichenko L, Kuznetsov S, Manolopoulos Y, editors. Data Analytics and Management in Data Intensive Domains. DAMDID/RCDL 2016. Communications in Computer and Information Science. Vol. 706. Cham: Springer; 2017

[26] Słoń G, Yastrebov A. Optimization and adaptation of dynamic models of fuzzy relational cognitive maps. In: Kuznetsov SO, Ślezak D, Hepting DH, Mirkin BG, editors. Rough Sets, Fuzzy Sets, Data Mining and Granular Computing. RSFDGrC 2011. Lecture Notes in Computer Science. Vol. 6743. Berlin, Heidelberg: Springer; 2011

[27] Meilinger T. The network of reference frames theory: A synthesis of graphs and cognitive maps. In: Freksa C, Newcombe NS, Gärdenfors P, Wölfl S, editors. Spatial Cognition VI. Learning, Reasoning, and Talking about Space. Spatial Cognition. Lecture Notes in Computer Science. Vol. 5248. Berlin, Heidelberg: Springer; 2008

[28] Solodilova AV. Cognitive Maps: Form of Representation of Scientific Knowledge of Historic. Minsk: BGPU; 2012

[29] Shulman Y. Towards a broadening of privacy decision-making models: The use of cognitive architectures. In: Hansen M, Kosta E, Nai-Fovino I, Fischer-Hübner S, editors. Privacy and Identity Management. The Smart Revolution. Privacy and Identity 2017. IFIP Advances in Information and Communication Technology. Vol. 526. Cham: Springer; 2018