UDC 004.8, 004.9

Genetic intelligence and tensor-unitary transformations

Petoukhov S.V. (Mechanical Engineering Research Institute of RAS)

The article is devoted to the author's results of the analysis of the genetically inherited ability of living bodies to intellectual activity (for example, the ability to echolocate in dolphins), which resulted in the emergence of algebraic formalisms called tensor-unitary transformations. Genetic intelligence is understood as that part of the intellectual potential of living organisms that allows, on the basis of genetic information in DNA and RNA molecules, to build, for example, from one fertilized cell an organism with trillions of cells so that the parental traits are reproduced in it in a multichannel noise-resistant manner, despite strong noise and constantly changing conditions of nutrition and external influences during life. In this case, we are talking about the systematic growth in the course of ontogenesis of the number of parameters and degrees of freedom of the body with a corresponding increase in the dimensionality of its configuration space of states. With such growth, the organism at successive stages of its development, acquiring new degrees of freedom and knowledge, somehow retains the memory of the skills and knowledge that it possessed at previous stages of life. The author develops the algebraic foundations for modeling this fundamental feature of the development of living bodies in the tensor-matrix language of systems of multidimensional vector configuration spaces. Tensor-unitary transformations are operators that preserve the lengths of vectors during their tensor transformation into vectors of a space of increased dimension (in contrast to conventional unitary transformations that transform vectors into a space of the same dimension). They are operators of expansion of stochastic-deterministic memory with preservation of all previous memory. Possible applications of tensor-unitary transformations for the development of AI, genetic algorithms, etc. are discussed.

Keywords: *genetic intelligence, development of organisms, configuration spaces, vector analysis, memory, stochastics and determinism.*

1. Introduction

This article is devoted to the analysis of genetically inherited ability of living bodies to perform intellectual activities (for example, the ability to echolocate in dolphins), as well as algebraic formalisms called tensor-unitary transformations. These transformations are proposed by the author as a result of the said analysis for modeling biological phenomena and possible development of approaches to artificial intelligence systems and new genetic algorithms.

In artificial intelligence systems, their creators strive to reproduce the properties of the natural intelligence of living bodies, the structures and mechanisms of which are inherited from generation to generation due to their connection with the genetic coding system. The author studies the structural features of the genetic coding system for the disclosure of information patents of living nature to ensure the inherited abilities of living organisms to perform intellectual activities.

Genetic intelligence is understood as that part of the intellectual potential of living organisms that allows, on the basis of genetic information in DNA and RNA molecules, to build, for example, from one fertilized cell an organism with trillions of cells in such a way that the parental characteristics are reproduced in it in a multichannel, noise-resistant manner, despite strong noise and constantly changing conditions of nutrition and external influences during life. In this case, we are talking about the systematic growth in the course of ontogenesis of the number of parameters and degrees of freedom of the body with a corresponding increase in the dimensionality of its configuration space of states. At the same time, with such growth, the organism at the next stages of its development, acquiring new degrees of freedom and knowledge, retains the memory of the skills and knowledge that it possessed at previous stages of life. The author develops the algebraic foundations for modeling this fundamental feature of the development of living bodies in the tensor-matrix language of systems of multidimensional vector configuration spaces. The desirable modeling tools should allow modeling the phenomenon of preserving the memory of past states in a developing vector system during its transition to new configuration spaces of increased dimension with the possibility of simultaneously modeling the expansion of memory with a corresponding increase in degrees of freedom.

In mathematical natural science, the tensor product of matrices is used to increase the dimensionality of the model vector space. The genetic coding system, unique in its speed properties, noise immunity and general biological significance, together with the family of DNA alphabets, is structured precisely for the tensor product of matrices, which links, for example, matrix representations of different DNA alphabets into a single tensor family of alphabetic matrices [10, 14]. The founder of quantum information science, Yu. I. Manin, introduced the concept of a quantum computer in his book [8] precisely when analyzing the features of high-speed processing of DNA information in chromosomes by “genetic automata”. He prophetically pointed out the important role of unitary rotations and tensor products: “A quantum automaton must be abstract: its mathematical model must use only the most general quantum principles, without prejudging physical implementations. Then the evolution model is a unitary rotation in a finite-dimensional Hilbert space, and the model of virtual division into subsystems corresponds to the decomposition

of space into a tensor product. Somewhere in this picture there must be a place for the interaction traditionally described by Hermitian operators and probabilities" [8, p. 15]. Thus, the very birth of quantum information science, so promising for the problems of artificial intelligence, occurred thanks to the desire to understand the features of genetic information science. Obviously, the universal rules of probabilities in genomic DNA as multilayered texts written in the languages of tensorial interconnected alphabets of n-plets of DNA, presented in this article, together with the unitary rotations tied to these rules, are consistent with this prediction of Yu. I. Manin (one example of the speed of genetic processes that amazed him is the process of replication of DNA strands in the bacterium E. coli at a speed of more than 1000 nucleotides per second [1].

.

2. Features of tensor-unitary transformations

The concept and apparatus of tensor-unitary transformations arose in the author as a result of studying the biological dualism of "stochastics-determinism", primarily in the information sequences of genomic DNAs of higher and lower organisms. Individual molecules interact in cells stochastically. In living bodies, everything is associated with stochastics. Even genetically identical cells of the same tissue have different levels of protein expression, sizes, etc. But with this stochastics in the "small", macro-corporeal traits are inherited from parents as determined. All genetics as a science began with Mendel's discovery of statistical rules for the inheritance of traits when crossing organisms. According to Mendel's law of independent inheritance of traits, information from the level of DNA molecules dictates the macrostructures of living bodies through many independent channels, despite strong noises. Thus, the colors of hair, eyes and skin are inherited independently of each other. Accordingly, each organism is a machine of multi-channel noise-resistant coding based on the dualism of stochastics-determinism.

The results presented in this article are based on the universal rules of statistical organization of information sequences of single-stranded genomic DNAs of higher and lower organisms, which the author discovered and which are related to the stochastic-deterministic dualism [11, 12]. It is logical to look for opportunities to model these universal rules based on the formalisms of quantum mechanics and quantum information science, since DNA belongs to the microworld of quantum mechanics, which is based on the concept of probabilities. In quantum mechanics and quantum information science, unitary transformations play an important role: the evolution of closed quantum systems is described by unitary transformations, and all calculations in quantum information science are based on unitary operators that play the role of logical gates (quantum

gates) [9]. Unitary transformations of vectors preserve their lengths and the values of scalar products (in the case of real components, they are orthogonal transformations).

By definition, tensor-unitary transformations are transformations that preserve the norms (lengths) of vectors during their tensor transformation into vectors of a space of increased dimension. Unlike unitary transformations, which transform the original n-dimensional vector into an n-dimensional vector of the same dimension, tensor-unitary transformations transform the original (or "parent") n-dimensional vector into a "daughter" m-dimensional vector of increased dimension (m>n) while preserving the length of the parent vector. We can say that they ensure the "inheritance" of the vector length into the length of a new tensor-transformed vector belonging to a space of tensor-increased dimension; this new space can be interpreted as the configuration space of a multiparameter system, the number of parameters of which has increased in the course of its development (by analogy with biosystems in the process of their ontogenetic growth).

Applying these tensor-unitary transformations to vectors can be thought of as a two-step process. First, the original n-dimensional vector, called the parent vector, is represented as the sum of all its basis vectors with their coordinate weights (e.g., the vector [x, y] is represented as the sum: x[1, 0]+y[0, 1]). Secondly, each of these weight basis vectors is tensor-multiplied by a so-called "norm-vector" (a qubit-like vector) that ensures that the length of the parent vector is preserved in the tensor-increased child vector. By definition, a norm-vector is a k-dimensional vector $[\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_{k-1}]$, the sum of the squares of the coordinates of which is equal to 1: $\alpha_0^2 + \alpha_1^2 + \alpha_2^2 + \ldots + \alpha_{k-1}^2 = 1$. Its coordinates can be interpreted as probability amplitudes (as in qubit or poly-qubit vectors), which in the general case can be either fixed numbers or functions of time or other variables. We briefly note that tensor-unitary transformations can be represented as a special case of Hadamard products for block vectors, in which we are talking about component-wise multiplication of a vector by the corresponding number of norm-vectors; a special symbol has been proposed to denote tensor-unitary transformations: ⍜ (its Unicode 0x235d, https://unicodemap.org/details/0x235D/index.html) [13].

In the case of a tensor-unitary transformation of the parent vector, the "qubit-likeness" of norm-vectors ensures the introduction of an element of probability (stochasticity) into the values of individual coordinates of the daughter multidimensional vector, special groupings of coordinates of which turn out to be deterministic carriers of the exact memory about the coordinates of the parent vector. Thus, tensor-unitary transformations, being stochastically-deterministic transformations, generate stochastically-deterministic vectors and allow modeling the biological dualism of "stochastics-determinism". Let us explain this with a simple example.

Let us consider the simplest 2-dimensional vector function [x(t), y(t)] = [x(t), 0] + [0, y(t)], and tensor multiply its first weight basis vector [x(t), 0] by the norm-vector [α_0, α_1], and its second weight basis vector [0, y(t)] by the norm-vector [β_0, β_1] (here $\alpha_0^2 + \alpha_1^2 = 1$, и $\beta_0^2 + \beta_1^2 = 1$). As a result, we obtain a 4-dimensional vector-function $\vec{\boldsymbol{D}}_1$:

$$\vec{\boldsymbol{D}}_1 = [x(t), 0] \otimes [\alpha_0, \alpha_1] + [0, y(t)] \otimes [\beta_0, \beta_1] = [x(t)\alpha_0, x(t)\alpha_1, y(t)\beta_0, y(t)\beta_1] \qquad (1)$$

The values of the individual coordinates of the daughter vector-function $\vec{\boldsymbol{D}}_1$ (1) are stochastic, since each of them contains one of the probability amplitudes α_0, α_1, β_0, β_1. There are infinitely many possible values of α_0, α_1, β_0, β_1, and their random selection changes the values of these individual coordinates, but the lengths of the child and parent vectors will be equal for all these selections and for each fixed value of the variable t. In other words, different sets of values α_0, α_1, β_0, β_1. give different daughter vector functions $\vec{\boldsymbol{D}}_1$, which differ from each other in their coordinates, but they all have the same length for any fixed value of the parameter t. This is confirmed by calculating the length $||\vec{\boldsymbol{D}}_1||$ of the vector-function $\vec{\boldsymbol{D}}_1$ (1) under any fixed value of the parameter t:

$$||\vec{\boldsymbol{D}}_1|| = \{x^2\alpha_0^2 + x^2\alpha_1^2 + y^2\beta_0^2 + y^2\beta_1^2\}^{0.5} = \{x^2(\alpha_0^2 + \alpha_1^2) + y^2(\beta_0^2 + \beta_1^2)\}^{0.5} = (x(t)^2 + y(t)^2)^{0.5} \qquad (2)$$

This length (2) of the daughter 4-dimensional vector function $\vec{\boldsymbol{D}}_1$ is equal to the length of the mother 2-dimensional vector function [x(t), y(t)] at any fixed time t. In other words, the daughter vector function $\vec{\boldsymbol{D}}_1$, having stochastic coordinates and increased dimensionality, retains an exact “memory” of the length of the mother vector function at any fixed time, regardless of the stochasticity of its individual coordinates, and in this respect is a deterministic entity. We can say that stochastics with hidden determinism takes place here. Now let us note other deterministic properties of this daughter 4-dimensional vector function $\vec{\boldsymbol{D}}_1$, associated with its projections onto coordinate planes.

A four-dimensional vector space with a Cartesian system of numbered coordinates [X_0, X_1, X_2, X_3] contains 6 coordinate planes:

$$(X_0,X_1),\ (X_2,X_3),\ (X_0,X_2),\ (X_0,X_3),\ (X_1,X_2),\ (X_1,X_3) \qquad (3)$$

In the plane (X_0,X_1) the daughter vector function $\vec{\boldsymbol{D}}_{\mathbf{1}}$ (1) is represented by its projection $\vec{\boldsymbol{M}}_{\mathbf{01}}$ = [x(t)α_0, x(t)α_1, 0, 0], the length of which $\|\vec{\boldsymbol{M}}_{\mathbf{01}}\|$ (4) is equal to the first coordinate of the parent vector function [x(t), y(t)] for any fixed t:

$$\|\vec{\boldsymbol{M}}_{\mathbf{01}}\| = \{x(t)^2\alpha_0^2 + x(t)^2\alpha_1^2\}^{0.5} = \{x(t)^2(\alpha_0^2 + \alpha_1^2)\}^{0.5} = x(t) \qquad (4)$$

In the plane (X_2,X_3) the daughter vector function $\vec{\boldsymbol{D}}_{\mathbf{1}}$ is represented by its projection $\vec{\boldsymbol{M}}_{\mathbf{23}}$ = [0, 0, y(t)β_0, y(t)β_1], the length of which $\|\vec{\boldsymbol{M}}_{\mathbf{23}}\|$ (5) is equal to the second coordinate of the mother vector function [x(t), y(t)] for any fixed t:

$$\|\vec{\boldsymbol{M}}_{\mathbf{23}}\| = \{y(t)^2\beta_0^2 + y(t)^2\beta_1^2\}^{0.5} = \{y(t)^2(\beta_0^2 + \beta_1^2)\}^{0.5} = y(t) \qquad (5)$$

Expressions (2, 4, 5) indicate that the daughter 4-dimensional vector-function $\vec{\boldsymbol{D}}_{\mathbf{1}}$, having stochastic coordinates, contains in the groupings of these coordinates the exact "memory" (or values) of all coordinates and the length of the parent 2-dimensional vector-function, i.e. in this respect it is a deterministic entity. Accordingly, tensor-unitary transformations are in this respect **the operators of stochastic-deterministic memory**

The simplest example considered shows that tensor-unitary transformations and the families of daughter vectors (or vector-functions) generated by them allow modeling the biological dualism of "stochastics-determinism", as well as the phenomena of Gestalt biology, in which only the aggregate values in groupings of elements are significant in contrast to the insignificance of individual elements in themselves, by analogy with the well-known genetically inherited phenomena of Gestalt psychology. For example, we recognize a musical melody even when it is performed on different instruments and in different frequency ranges with a change in the absolute values of the sound frequencies of individual notes, which turn out to be insignificant in contrast to the ratios in groupings of note frequencies. Since tensor-unitary transformations allow modeling biological Gestalt phenomena, they can also be conventionally called Gestalt transformations, and the stochastic-deterministic vectors (or vector functions) generated by them can be called Gestalt-vectors.

It should be emphasized that usually unitary (or orthogonal) operators are understood as unitary square matrices, but in the case of tensor-unitary transformations, the component-wise tensor product of the original vector by the corresponding number of norm-vectors is used to obtain

a stochastic-deterministic object. In the general case, the number of norm-vectors is equal to the dimension of the parent vector (or the daughter vector of the previous generation), each component of which must be multiplied by a separate norm-vector.

Note that the daughter 4-dimensional vector $\vec{\boldsymbol{D}}_{\mathbf{1}}$ (1) is not only a carrier of the exact memory of the coordinates of the parent 2-dimensional vector, but also a carrier of new information in the planes (X_0, X_2), (X_0, X_3), (X_1, X_2), (X_1, X_3) of the 4-dimensional space. In these planes, the vector $\vec{\boldsymbol{D}}_{\mathbf{1}}$ is represented by its projections $\vec{\boldsymbol{M}}_{\mathbf{02}} = [x(t)\alpha_0, 0, y(t)\beta_0, 0]$, $\vec{\boldsymbol{M}}_{\mathbf{03}} = [x(t)\alpha_0, 0, 0, y(t)\beta_1]$, $\vec{\boldsymbol{M}}_{\mathbf{12}} = [0, x(t)\alpha_1, y(t)\beta_0, 0]$, $\vec{\boldsymbol{M}}_{\mathbf{13}} = [0, x(t)\alpha_1, 0, y(t)\beta_1]$, the lengths of which are not equal to the coordinates of the maternal vector.

The values of their lengths carry new information, which is introduced into them by the stochastic coordinates of the norm-vectors α_0, α_1, β_0, β_1. These new stochastic components can reflect, for example, the specificity of the ontogenesis stage or the impact of the external environment on the organism. Thus, the tensor-unitary transformation is an **operation of memory expansion** (with the preservation of all previous memory) for a tensor-extended multiparameter system, which, along with the memory of the parameters of the "ancestors", contains many new parameters or information. Accordingly, tensor-unitary transformations can be used to model growing stochastic-deterministic systems of morphogenetic, biorhythmic and other types, where the number of parameters and degrees of freedom increases step by step.

The values of the scalar products of two parent vectors and their two daughter vectors are equal only when the tensor-unitary transformations of both parent vectors use the same set of norm-vectors.

Tensor-unitary transformations can be repeatedly applied to develop from a parent vector increasingly complex multiparameter child vectors with a step-by-step increase in the dimensionality of their configuration spaces. In certain groupings of coordinates of these growing child vectors (i.e. in projections onto subspaces of the configuration space), information about all coordinates of both parent vectors and previous child vectors will be preserved, and new information will also be presented (in projections onto other subspaces). Tensor-unitary transformations allow one to record the step-by-step history of the development of a tensor-growing multiparameter stochastic-deterministic system in the form of sequences of daughter vectors corresponding to the original parent vector and the norm-vectors used.

Let us repeat that the coordinates of norm vectors can be not only fixed values of probability amplitudes, but also normalized functions of time or other variables. Expression (6) shows an

example of two-dimensional norm-vectors [α_0, α_1], the coordinates of which are functions of the variable t, and the sum of the squares of the coordinates is equal to 1:

$$\overrightarrow{\boldsymbol{F(t)}} = [\alpha_0(t)/\{\alpha_0(t)^2 + \alpha_1(t)^2\}^{0.5}, \alpha_1(t)/\{\alpha_0(t)^2 + \alpha_1(t)^2\}^{0.5}] \quad (6)$$

At any value *t*, the sum of squares of its coordinates is equal to 1, that is, the vector function $\overrightarrow{\boldsymbol{F(t)}}$ satisfies the definition of normvectors and can be used in tensor-unitary transformations. The maternal vector can also be normalized, that is, have coordinates whose sum of squares is equal to 1. An example of such a vector:

$$[x(t)/\{x(t)^2 + y(t)^2\}^{0.5}, \quad y(t)]/\{x(t)^2 + y(t)^2\}^{0.5}] \quad (7)$$

The tensor-unitary product of such a normalized parent vector by a norm-vector yields a vector, the sum of the squares of the coordinates is again equal to one, i.e. it again generates a norm-vector. This allows one to model **multiple reproductions** of parametrically defined geometric configurations using tensor-unitary transformations

For modeling bio-cyclic phenomena, the case when the functions serving as coordinates of the normalized parent vector of type (7) are cyclic, for example, consist of superpositions of sines and cosines, is of particular interest. Then the daughter vectors generated by tensor-unitary transformations will also be endowed with coordinates of a cyclic nature for modeling multiparameter systems consisting of many subsystems of cyclic behavior. The importance of this case is due to the fact that the organism is a huge chorus of coordinated cyclic processes, the number of which increases as it develops ontogenetically from the embryonic to the mature state, with step by step obtaining new and new degrees of freedom, coordinated with the already existing ones. For example, in the organism of an adult, their number reaches enormous values, since it contains approximately 100 trillion cells participating in these sets of coordinated cycles. Moreover, all these coordinated cyclic processes occur against the background of random (stochastic) interactions between individual molecules in cells and are themselves stochastically determined to a certain extent. According to the provisions of ancient chrono-medicine, all our diseases are the result of a violation of this coordination.

Note that tensor-unitary transformations can be applied not only in the case of real numbers, but also in the cases of various systems of multidimensional numbers: complex numbers, hyperbolic numbers, Hamilton quaternions, Cockle split-quaternions, and so on.

So far we have been talking about tensor-unitary modeling of growing multiparametric systems (such as multicellular biosystems in ontogenesis). But we can also talk about similar tensor-unitary modeling of such families of computer memory cells that are gradually expanded in the course of their model development. This makes it easier to understand that tensor-unitary transformations can lead to computer artificial intelligence of the genomorphic type: such intelligence can be based on generations of tensor-unitary expanding families of computer memory cells, in which some groups of cells carry information about the states of all cells of families of previous generations, and other groups of cells carry new information related to the current effects of the external and internal environment.

For this reason, tensor-unitary transformations and the families of generations of stochastic-deterministic vectors generated by them seem to be promising algebraic tools for creating genomorphic-type artificial intelligence systems. They are also useful for developing genetic algorithms that use the principle of natural selection to “grow” artificial intelligence and are used worldwide on the basis of hundreds of patents for many technical problems [2]. The listed algebraic properties of tensor-unitary transformations and the families of generations of daughter multidimensional vector functions generated by them partially clarify why the universal rules of stochastic organization of genomic DNA are structured by nature in accordance with these algebraic operations and families.

In quantum mechanics, unitary transformations describe the evolution of closed quantum systems. The author believes that tensor-unitary transformations are useful for developing the quantum mechanics of open quantum systems, growing with the increase of their multicomponent composition similar to biological bodies developing during ontogenesis and phylogenesis. They are also associated with the tensor-matrix theory of digital antenna arrays and the doctrine of bio-antenna arrays as the basis of energy-information biological evolution [12]. A few additional words should be said here. In our time, special attention is paid to the possibilities of using in evolutionary biology the achievements of mathematical natural science and informatics, for example, from the fields of noise-immune coding and information transmission, physical fields theory, holography, quantum informatics, etc. One of the rapidly developing scientific and technical areas is the theory of digital antenna arrays, which has extensive applications: medical ultrasound scanning technology (on multichannel platforms with digital emitter arrays), sonar systems, seismographs, meteorological instruments, radio relay stations, avionics, radio astronomic devices, etc. The formation of these applications is accompanied by the intensive development of new computational methods.

Antenna arrays coordinately combine many individual antennas into a single system - from a few antennas to many thousands of antennas. The emergent properties of such systems provide their amazing functionality, which far exceeds the capabilities of individual antennas and causes humanity to saturate and envelop the Earth with millions of antenna arrays. Such a combined array of antennas (technical or biological) can grow and expand depending on the specific conditions of its existence, being replenished by connecting new sets of antennas to it. The spatial configuration and dimensions of antenna arrays can be very different, but they all operate on a matched emission and reception of electromagnetic and other waves by separate antennas in their composition. The chemical and structural composition of antenna arrays can also be different and include, among other things, photonic crystals and liquid crystals, examples of which are abundant in living bodies. Modern science sees great prospects with nanoantennas, which are expected to lead to revolutionary changes in computer technology (photonics) and energy (efficient use of solar energy). Nanoantennas based on DNA are already used in scientific technologies: Canadian scientists have created glowing nanoantennas from DNA molecules to track the relationships within proteins. These nanoantennas are capable of fluorescence and can absorb radiation at one wavelength and emit light at a different frequency depending on the molecular environment [5]. This antenna is 5 nm long and is the smallest antenna ever made. It can be assumed that humanity is entering the era of the technological use of biomolecular antennas.

The importance of antenna arrays in different technical fields has led to the intensive development of the mathematical theory of transmitting and receiving antenna arrays of various types, which is presented in many publications (look at review in [12]. The mathematical description of the operation of engineering antenna arrays, operating to emit or receive waves, is almost the same. The concept of antenna arrays with its special computational methods is essential for the concept of biological computation since electromagnetic waves are capable of transmitting information in the course of biological computing, as noted in the works [6, 7]. The topic of antenna arrays is important for algebraic modeling and understanding the mechanisms of genetic intelligence, which ensures the step-by-step development of a genetically encoded organism from a single fertilized cell into a single colony of trillions of mutually coordinated cells. The author's doctrine about bio-antenna arrays as the basis of energy-information biological evolution gives new approaches to study some inherited possibilities of living bodies for operations of intellectual types such as, for example, echolocations of dolphins and bats.

But in the scientific literature (before the author’s publications), it was not possible to find a single publication that would connect biological phenomena with the emergent properties of

antenna arrays. Filling this gap, the author has studied and identified many heritable biological structures associated with the idea of bio-antenna arrays and their wave activity [12]. It seems natural to assume that biological evolution has not passed by the emergent properties of antenna arrays, which have conquered the modern technology of remote interconnection and sense of the surrounding space. One can believe that tensor-unitary transformations will be useful for developing algebraic approaches in this field.

3. 3. Stochastic determinism as an antipode to deterministic chaos

Our world is full of random events. The life and development of genetically inherited bio-bodies are inextricably linked with random interactions of molecules, against which the inherited deterministic macrostructures with the parental traits are realized. So, hidden determinism and the corresponding laws of stochastic determinism are hidden behind these accidents. The important role of probabilities in Nature is reflected in quantum mechanics based on the concept of probabilities. At the same time, as it is well-known, the Nobel laureate in physics R. Feynman noted that nobody really understands quantum mechanics. The studies of the biological dualism “stochastics-determinism” and the universal rules of the stochastic organization of genomic DNAs, presented in this paper, draw attention to the need to develop the theory of “stochastic determinism” (or “chaotic determinism”) as an antipode to the well-known theory of “deterministic chaos”. One can believe that in the future theory of stochastic determinism, a prominent place will be occupied by the tensor-unitary transformations described above, which model the named dualism, biological gestalt phenomena, and tensor reproduction of biostructures. Let's look at the differences between these two theories.

The theory of deterministic chaos (or dynamic chaos) has been developed by the works of a large number of mathematicians and physicists. It has been the subject of a large number of publications. Deterministic chaos is a phenomenon and a part in the theory of dynamical systems, in which the behavior of a nonlinear system looks random, despite the fact that it is determined by deterministic laws. The reason of this phenomenon is instability (sensitivity) with respect to the initial conditions and parameters: a small change in the initial condition over time leads to arbitrarily large changes in the dynamics of the system. Dynamics that is sensitive to the slightest changes in the initial conditions of the system, from which its development, change begins, and in which these slightest deviations multiply many times over time, making it difficult to predict the

future states of the system, is often called chaotic. A well-known example of a system of deterministic chaos is Sinai billiards [3].

What is the difference between stochastic determinism, represented in the phenomena of biological dualism "stochastics-determinism", and deterministic chaos? Table 1 shows the main differences.

Table 1. The main differences between deterministic chaos and stochastic determinism

	Deterministic chaos	**Stochastic determinism**
	The birth of the random from the non-random is observed. A completely deterministic system generates unpredictable states that have the properties of a random process.	The birth of the deterministic (non-random) from the random is observed. A system characterized by random interactions of molecules at the cellular level gives rise to deterministic biological forms.
	A small change in the initial condition leads to arbitrarily large changes in the dynamics of the system over time (it is instability or sensitivity of the system with respect to the initial conditions).	Random interactions among molecules at the beginning of the biosystem development have little effect on the final deterministic result of its development (stability or insensitivity with respect to initial conditions).
	In systems of deterministic chaos, it is customary to consider multi-parameter systems with a fixed dimension of their configuration spaces (for example, Sinai billiards).	In the stochastic determinism of biological systems, systems are considered with tensor development of the dimension of their configuration spaces (for example, when sets of interconnected billiards of Sinai are born tensorically in the course of development).

One of the distinguishing features of the living is the presence of forms of constant movement in it. No wonder they say that "life is movement". Correspondingly, a living cell can always be distinguished from a dead one by this feature: in almost all biological tissues, cells move

continuously, albeit slowly, or at least change shape. But it is not at all a thermal, Brownian motion. On the contrary, "*Brownian motion in a eukaryotic cell is a sign of its death"* [4, p. 7]. Taking into account all having materials about biological dualism "stochastics-determinism", the author puts forward the following hypothesis: in living bodies, a special type of stochastic (or stochastic-deterministic) motions exist, the study of which is important for understanding biological stochastic-deterministic phenomena. It is probably that this kind of motions has its own principles of minimization and conservation laws. The algebraic toolkit described above can be useful in these future studies and in developing the theory of stochastic determinism as well. The development of the algebraic theory of stochastic determinism using tensor-unitary transformations seems useful for the formation of new approaches to genomorphic-type artificial intelligence systems based on the fundamental organization principles of genetic coded biological bodies.

4. Conclusion

The tensor-unitary transformations proposed by the author seem to be a useful tool for developing model approaches to studying genetic intelligence, coordinated growth of parts in developing multiparameter biological and technical systems, as well as the development of genetic algorithms in connection with the creation of AI. The topic of genetic intelligence is also connected with the doctrine of bio-antenna arrays as the basis of energy-information biological evolution [12].

Acknowledgments

The author thanks S. Ya. Kotkovsky, Yu. I. Manin, V. I. Svirin, I. V. Stepanyan, and G. K. Tolokonnikov for useful discussions on the topic of tensor-unitary transformations and their applications.

References

1. Bank E. How Much Time Does It Take for a DNA Molecule to Replicate? - sciencing.com, 2022, https://sciencing.com/much-time-dna-molecule-replicate-21660.html
2. Buontempo F. Genetic algorithms and machine learning for programmers: create AI models and evolve solutions // Pragmatic Bookshelf. 2019. 236 p. ISBN-10 : 168050620X, ISBN-13: 978-1680506204.
3. Dorfman J. R. An Introduction to Chaos in Nonequilibrium Statistical Mechanics // Cambridge, England: Cambridge University Press, 1999.
4. Fulton A. Cellular architecture and choreography // London, Chapman and Hall, 1984.
5. Harroun S. G., Lauzon D., Ebert M., Desrosiers A., Wang X. and Vallée-Bélisle A. Monitoring protein conformational changes using fluorescent nanoantennas. // Nature

Methods. 2000. Vol. 19, P. 71-80. (https://doi.org/10.1038/s41592-021-01355-5).

6. Igamberdiev A. U., Shklovskiy-Kordi N. E. The quantum basis of spatiotemporality in perception and consciousness // Progress in Biophysics and Molecular Biology. 2017. Vol. 130, P. 15-25.
7. Liberman E. A. Cell as a molecular computer // Biofizika. 1972. Vol. 17, P. 932-943.
8. Manin Yu. I. Computable and non-computable // M., Soviet Radio. 1980 (in Russian).
9. Nielsen M. A., Chuang I. L. Quantum Computation and Quantum Information // New York: Cambridge Univ. Press, 2010.
10. Petoukhov S.V. Matrix genetics, algebras of the genetic code, noise-immunity // Moscow, Regular and Chaotic Dynamics. 2008. 316 p. (in Russian), http://petoukhov.com/matrix-genetics-petoukhov-2008.pdf .
11. Petoukhov S.V. Algebraic Rules for the Percentage Composition of Oligomers in Genomes // Preprints 2021, 2021010360, DOI: 10.20944/preprints202101.0360.v3, 2021.
12. Petoukhov S.V. The stochastic organization of genomes and the doctrine of energy-information evolution based on bio-antenna arrays // Biosystems, 104712, 2022a. ISSN 0303-2647, https://doi.org/10.1016/j.biosystems.2022.104712.
13. Petoukhov S.V. Rules of Stochastics of Genomic DNAs, Biological Dualism "Stochastics-Determinism", and Tensor-Unitary Transformations // Preprints 2022, 2022080435, 2022b (doi: 10.20944/preprints202208.0435.v1).
14. Petoukhov S.V., He M. Algebraic biology, matrix genetics, and genetic intelligence // Singapore, World Scientific, 2023, 616 p., https://doi.org/10.1142/13468.

UDC 512.581.2, 004.032.26

Направленные бинарные категорные склейки, принцип двойственности и категорная модель нейронных сетей

Толоконников Г.К. (ФНАЦ ВИМ РАН)

Теория категорных систем, развиваемая автором, позволяет естественным образом смоделировать традиционные искусственные нейронные сети произвольной топологии, сети живых нейронов, у которых помимо спайковой коммуникации имеется несколько десятков других видов клеточной коммуникации, а также сетевые структуры, аналогичные высшим категориям. Математическим аппаратом категорных систем является теория категорных склеек, данная работа посвящена следующим вопросам этой теории. Вводятся и изучаются направленные бинарные категорные склейки, являющиеся обобщением обычных категорий, в теории которых, как известно, большую роль играет понятие дуальности категорий и принцип двойственности, основанный на дуальности. В теории категорных склеек помимо дуальности аналогичной дуальности, порождаемой заменой направления стрелок категории, имеется новый вид дуальности, связанный с заменой имен стрелок на имена свёрток, обобщающих обычную операцию композиции. Проведено построение дуальных в обоих смыслах категорных склеек изучаемого типа, доказаны теоремы, отвечающие принципам двойственности для указанных двух видов дуальности. Теоремы приведены в рамках теории доказательств, для частного случая обычных категорий дающие новые доказательства принципа двойственности. Обобщение подхода на свёрточные аналоги мультикатегорий находят приложения в нейронных сетях, в частности, для известных формул С. Осовского в методе обратного распространения ошибки.

Keywords: *нейронные сети, категории, мультикатегории, поликатегории, свёрточные поликатегории, категорные склейки, системы, функциональные системы, категорные системы, системообразующий фактор, дуальность, принцип двойственности*

1. Введение

Теоретико-множественной парадигмой мы будем называть положение о том, что объекты физического и ментального мира моделируются с помощью множеств и подмножеств. Известный многотомный курс Николя Бурбаки формализовал теоретико-множественную парадигму. Мы используем системную парадигму, когда множества и подмножества

заменены на системы и их совокупности. Понятие «множество» имеет интуитивную интерпретацию в наивной теории множеств. Наивная теория множеств формализуется в виде аксиоматических теорий множеств.

Мы используем понятие категорной системы в качестве системы. Мы предложили формализацию интуитивного понятия системы в рамках категорной теории систем.

Интуитивное понятие системы, которое мы используем, обобщает функциональные системы Петра Кузьмича Анохина, открытые им в двадцатые годы прошлого века. Отметим, что приоритет открытия ключевых положений кибернетики (обратная связь и тому подобное) Норберт Винер публично признал за Петром Кузьмичом Анохиным будучи в Москве на конгрессе в 1960 году, поскольку эти положения содержались в теории функциональных систем (почти за 20 лет до выхода известной книги Винера по кибернетике).

Основным открытием Петра Кузьмича Анохина в теории систем является системообразующий фактор, трактуемый им в физиологии, как полезный результат для организма.

Функциональная система трактуется как инструмент, переводящий организм из имеющегося состояния в состояние с достигнутым полезным результатом. Подобному определению функциональной системы соответствует понятие стрелки в теории категорий с началом в виде исходного состояния и концом в виде состояния с достигнутым полезным результатом. Адекватным языком при таком подходе к системам является язык теории категорий.

Категорная природа функциональных систем вскрыта в категорной теории систем, где системообразующий фактор обобщён для случаев, включающих неживые и другие системы.

Согласно системной парадигме функциональная система и категорная система формируются из других систем, которые становятся подсистемами формируемой системы. Соединение систем в формируемую систему в категорной теории систем осуществляется с помощью операции свёртки.

Важным положением теории функциональных систем, ставшим в теории категорных систем постулатом, является требование выводить все свойства системы, исходя из её системообразующего фактора. Описание свойств системы проводится в рамках теории объекта, который является данной системой, указанные требования приводят к необходимости для теории каждой системы вырабатывать, в частности, алфавит, язык, понятие высказывания, конструктивные операции, понятие истинности высказываний,

логику, исходя из системообразующего фактора. По поводу сказанного за деталями можно обратиться [1] к докладу автора на конференции MathAI-2025 (https://siriusmathcenter.ru/055w) и имеющимся там ссылкам. Отметим, что подобное системное построение теории в значительной степени проведено Андреем Андреевичем Марковым для теории слов в алфавитах в его книге [2].

Известное определение Месаровича [3] задаёт систему в рамках теоретико-множественной парадигмы в виде отношения на декартовом произведении множества входов и множества выходов (чёрный ящик). Категорным обобщением системы по Месаровичу является полистрелка поликатегории. Поликатегории ввёл Сабо в 1975 году. Соединение полистрелок осуществляется обычными композициями, имеющими, как показал Гарнер [4], ограниченные возможности. Автор заменил композиции на операцию свёрток, возникли свёрточные поликатегории, которые в отличие от поликатегорий Сабо позволяют, например, моделировать спайковые соединения нейронов, в том числе, моделировать искусственные нейронные сети произвольной топологии. В дальнейшем теория свёрточных поликатегорий была расширена автором до теории категорных склеек, в которой оказалось возможным моделировать неспайковые связи нейронов и других клеток, а также аналоги нейронных сетей, отвечающих высшим категориям.

В искусственных нейронных сетях отдельные нейроны соединяются друг с другом так, что с выхода аксона нейрона путь сигнала (пунктир на рис. 1) разветвляется для соединения нейрона с несколькими другими нейронами, при этом возникает точка ветвления (синяя точка на рис. 1). Точка ветвления с пунктирными линиями отвечает указанной свёртке в категорной модели нейросети. Непосредственно видно, что свертка напоминает нейрон. Это наблюдение на интуитивном уровне отвечает исследуемому в работе новому виду дуальности («между» мультистрелками и свертками) в мультикатегориях.

Нижняя часть рисунка 1 изображает нейронную сеть с нейронами, имеющими один вход и один выход, моделируемую в нашем подходе направленными бинарными склейками.

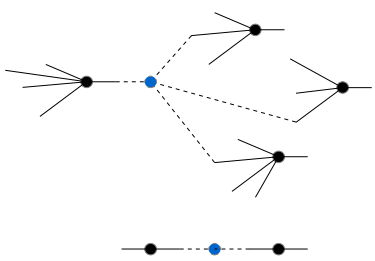


Рис.1. Соединение в искусственной нейросети нейронов свёрткой (пунктир)

В настоящей работе мы проведём полное построение с выделением конструктивной части основ теории бинарных направленных склеек, частным случаем которых являются обычные категории. В частности, будет детально описан новый вид дуальности, имеющей место для бинарных склеек, помимо дуальности (двойственности), отвечающей смене направления стрелок в обычных категориях, и дано доказательство расширенного принципа двойственности, аналогичного имеющемуся принципу двойственности теории категорий [5].

Аналогичные результаты имеют место для свёрточных мультикатегорий (моделирующих нейронные сети произвольной топологии), их детальное изложение отложено до отдельной статьи.

С точки зрения категорной теории систем естественным образом выделяется (что сделано в работе для теории бинарных склеек) конструктивная часть теории (общесистемная конструктивность, см. [1]), к которой могут быть добавлены различные логики. Здесь мы касаемся классической логики, интуиционистской логики, приводим ряд замечаний по конструктивной логике А.А.Маркова [2], изложение ведется в рамках теории доказательств. Тематика статьи носит весьма междисциплинарный характер, поэтому желательно достаточно подробное изложение, чему уделено внимание в статье (приведены некоторые несложные выкладки, деревья выводов и т.п.).

.

2. Категорные склейки

Рассмотрим следующий язык, имеющий 8 сортов. Мы не фиксируем заранее логику, оставаясь в рамках конструктивного подхода, с конструктивной операцией приписывания букв алфавита к имеющемуся слову. Алфавит имеет переменные и, соответственно, функциональные символы проекций $x_m^{(i)}, C^{(j)}$, *i, j*=1,2,..., 8*, m* - натуральное число, выбор *i* или *j* означает выбор сорта переменной. Обычным образом вводим в левой польской записи (без использования скобок при построении термов и формул) термы $t^{(i)}, v^{(j)}$, ... построенные из указанных переменных и проекций, отмечая явно сорт терма верхним индексом.

Введём двуместные буквы равенства для каждого сорта $=_i$, при этом некоторые равенства $w^{(j)}=_j v^{(j)}$ определяют оговариваемые явно конструктивные операции, используемые для построения формул гребней категорных склеек.

Иногда для наглядности мы выписываем выражения с равенствами обычным образом, в этих случаях их польская запись легко восстанавливается.

Вводим функциональные символы ξ_{ij} вида $(i \to j)$ со свойствами (аксиомами)

$$\xi_{ij}\xi_{ji}t^{(i)}=_i t^{(i)}, i\neq j \quad . \qquad (1)$$

Введём также равенства между сортами, как обозначения $t^{(i)}=_{ij} v^{(j)} \stackrel{\text{def}}{=} t^{(i)}=_i \xi_{ij} v^{(j)}$.

Базовыми формулами назовём выражения вида

$$t^{(i)}=_{ij} v^{(j)} \quad . \qquad (2)$$

Некоторые явно выписываемые формулы мы называем ***аксиомами***. Аксиомы определяют используемые для построения формул гребней склеек конструктивные операции.

Для каждого *i* обозначим $C_i^{(j)} \stackrel{\text{def}}{=} \xi_{ij} C^{(j)} \xi_{ji}$ и вводим аксиомы

$$C_i^{(k)} C_i^{(j)} t^{(i)} =_i C_i^{(l)} t^{(i)} \quad .$$

Таким образом, функциональные символы образуют алгебру, с таблицей умножения, отвечающей последним аксиомам.

Для бинарных склеек по определению эта алгебра имеет вид

$$C_i^{(k)} C_i^{(j)} t^{(i)} =_i C_i^{(j)} t^{(i)} \quad . \qquad (3)$$

Данные равенства задают по определению конструктивные операции, называемые ***правилами приведения проекций***: терм $C_i^{(k)} C_i^{(j)} t^{(i)}$ заменяется на терм $C_i^{(j)} t^{(i)}$. Конструктивные операции со словамим обычно можно записать в виде нормальных алгорифмов А.А.Маркова, так, правило приведения проекций можно задать в виде алгорифма с одной формулой подстановки $C_i^{(k)} C_i^{(j)} \to C_i^{(j)}$.

Для построения гребней категорных склеек постулируем применение следующего ***правила приведения равенств к проекциям***:

$x_i = C\, x_j$ заменяется на формулу $C\, x_i = C\, x_j$.

Это правило согласуется с предыдущими аксиомами алгебры проекций.

Вводим в алфавит знак конъюнкции $\wedge$, как бинарный функциональный знак на формулах. ***Формулами*** называем выражения, получаемые применением конъюнкции к базовам формулам и любым выражениям полученным таким применением. Чтобы охватить конструктивный случай, мы не используем без отдельных оговорок понятие множества (совокупности) всех формул и других понятий, где необходимо обращаться к актуальной бесконечности.

Постулируем свойство ***ассоциативности конъюнкции***, что позволяет не использовать скобок при обычной записи конъюнкций. Точнее, вводится конструктивная операция, используемая при построении формул гребней склеек, ***приведения коньюкций***, формула $\wedge\wedge fgh$ переходит в формулу $f\wedge g\wedge h$, а формула $\wedge f\wedge gh$ переходит в формулу $f\wedge g\wedge h$.

Вводим также переменные $a_k^{(i)}$ со свойствами

$$a_k^{(i)} = C^{(i)} x_k^{(i)} \quad . \qquad (4)$$

Расширим правило приведения проекций, добавив в него правило замены, задаваемой формулой подстановки $C^{(j)} a_k^{(i)} \to a_k^{(i)}$.

Вводим для каждого сорта набор функциональных символов $\mu_l^{(i)}, \ldots, l = 1,2, \ldots$, как символов *n*-арных операций, *n*=0,1,2,3,... , расширяем понятие терма, включив указанные операции в процедуру построения термов.

Категория является совокупностью стрелок (если объекты отождествлять, как это часто делается, с единичными стрелками), имеющими наглядный образ направленных отрезков. Аналогично категорная склейка имеет в качестве элементов изображаемые наглядно ряды вертикальных чёрточек, которые мы будем называть ***гребнями***, как и соответствующие им формулы. Определяемый ниже гребень категорной склейки состоит из внешнего гребня, внешней свёртки, внутреннего гребня и внутренней свёртки.

Для введённых 8 сортов используем следующие обозначения (z - один из символов, имеющих сорт):

$$z^{(3)} = z'^{(1)}, z^{(4)} = z'^{(2)}, z^{(5)} = \overline{z}^{(1)}, z^{(6)} = \overline{z}^{(2)}, z^{(7)} = \overline{z}'^{(1)}, z^{(8)} = \overline{z}'^{(2)}.$$

Используя коммутативность и ассоциативность конъюнкции, формулу в языке можно выписать в 2x2-таблице, считая формулы в клетках таблицы соединенными не указываемыми в такой записи конъюнкциями, переход к такой записи формулы является ***конструктивной операцией***.

В ячейку (1,1) размещаем нештрихованные и без надчеркивания букв подформулы ***внешних гребней.*** В ячейку (1,2) размещаем содержащие буквы со штрихами подформулы ***внешних свёрток.*** В ячейку (2,1) размещаем содержащие буквы с надчеркиванием подформулы ***внутренних гребней*** и в ячейку (2,2) размещаем содержащие буквы с надчеркиванием и штрихами подформулы ***внутренних свёрток***:

u	v'
$\overline{w}$	$\overline{r'}$

При размещении формулы в таблице штрихов и надчеркиваний можно не указывать, они при необходимости восстанавливаются очевидным образом.

Для сортов имеем

1, 2	3, 4
5, 6	7, 8

или мнемонически в обозначениях, заданных выше

1, 2	1', 2'
$\overline{1}$, $\overline{2}$	$\overline{1'}$, $\overline{2'}$

Мы, не имея пока никакой логики, не задаёмся вопросом (и не делаем подобных утверждений) всякая ли формула может быть приведена к виду таблицы (или ко всякой ли формуле может быть применена данная конструктивная операция).

Определение

Определим конструктивные ***операции проекций*** P_{ij}, i,j=1,2, которые переводят исходный гребень, в результирующий гребень, заменой на знак пустого множества всех формул в клетках, отличных от клетки (i,j).

Определение

Полным базовым гребнем называется формула, составленная из конъюнкций формул (i, j = 1, 2)

$$a_k^{(i)} = {}_i C^{(i)} x_k^{(i)},\ a'^{(i)}_k = {}'_i C'^{(i)} x'^{(i)}_k,$$
$$\bar{a}'^{(i)}_k = {}'^{-}_i \bar{C}'^{(i)} \bar{x}'^{(i)}_k,\ \bar{a}_k^{(i)} = {}^{-}_i \bar{C}^{(i)} \bar{x}_k^{(i)}$$

и равенств $\eta_k^{(i)} = \theta_m^{(j)}$, $\eta_k^{(i)}, \theta_m^{(j)}$ могут быть любыми из символов $a_n^{()}, x_m^{()}, n, m = 1,2,\ldots$, в последних формулах знак равенства имеет соответствующие опущенные здесь индексы. При этом построение базового гребня включает правило приведения равенств к проекциям, в результате равенства в базовых гребнях содержат либо переменные, либо их проекции.

По построению формул полных базовых гребней подформулы равенства содержат только те буквы переменных, которые встретились в подформулах вида $a = Cx$. Это позволяет знаки равенства (формулы с равенствами переменных) исключить из таблицы, заменив формулы равенства пунктирами, соединяющими буквы переменных, входящие в равенства.

Определение

Базовые гребни, переменные $x_k^{(i)}$ которых входят в равенства (соединены пунктиром) называются ***связными гребнями***.

По построению гребни состоят из связных гребней.

Для случая направленных бинарных склеек построение гребней проводится, начиная с некоторых внешних базовых гребней с помощью операций свёртки.

Определение

Внешними базовыми связными ***гребнями*** направленных бинарных склеек называются следующие формулы

$$a_k^{(1)} = C^{(1)} x_k^{(1)} \wedge a_m^{(2)} = C^{(2)} x_m^{(2)} \wedge x_k^{(1)} =_{1,2} x_m^{(1)}$$

или

$$a_k^{(1)} = C^{(1)} x_k^{(1)} \wedge a_m^{(2)} = C^{(2)} x_m^{(2)} \quad .$$

Переменные $a_k^{(1)}$ называются ***входами***, переменные $a_k^{(2)}$ называются ***выходами***, (отсюда в названии склейки взят термин «направленный») алгебра проекций имеет вид (нижний индекс у C опущен):

$$C^{(1)} C^{(2)} t = C^{(2)} t \, , C^{(2)} C^{(1)} t = C^{(1)} t \, , C^{(1)} C^{(1)} t = C^{(1)} t \, , C^{(2)} C^{(2)} t = C^{(2)} t \quad . \qquad (5)$$

Таким образом, гребень в изучаемом случае состоит всего из двух «зубъев».

Определение

Внешними базовыми связными ***свёртками*** направленных бинарных склеек называются следующие формулы

$$a'^{(1)}_k = C'^{(1)} x'^{(1)}_k \wedge a'^{(2)}_m = C'^{(2)} x'^{(2)}_m \wedge x'^{(1)}_k = {}'_{1,2} x'^{(2)}_m$$

или

$$a'^{(1)}_k = C'^{(1)} x'^{(1)}_k \wedge a'^{(2)}_m = C'^{(2)} x'^{(2)}_m \quad .$$

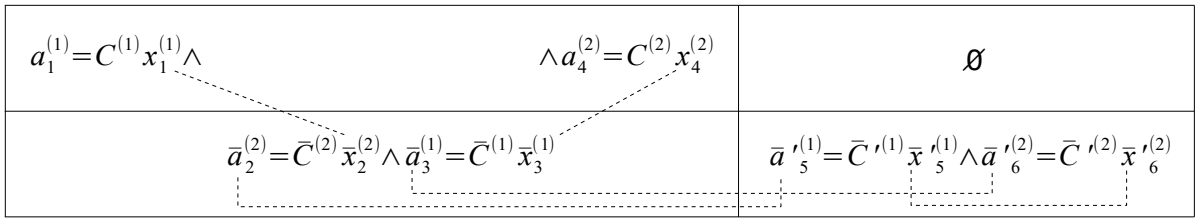
Также переменные $a'^{(1)}_k$ называются ***входами*** (свёрток), переменные $a'^{(2)}_k$ называются ***выходами*** (свёрток)**.**

Из введённых гребней и свёрток строятся, как это показано далее, остальные возможные полные гребни направленных бинарных склеек.

Рассмотрим следующий полный гребень, внешний гребень которого состоит из двух связных гребней, и имеющий равенства $a_2^{(2)} = a'^{(1)}_5 \, , a_3^{(1)} = a'^{(2)}_6$

$a_1^{(1)} = C^{(1)} x_1^{(1)} \wedge a_2^{(2)} = C^{(2)} x_2^{(2)} \wedge a_3^{(1)} = C^{(1)} x_3^{(1)} \wedge a_4^{(2)} = C^{(2)} x_4^{(2)}$	$a'^{(1)}_5 = C'^{(1)} x'^{(1)}_5 \wedge a'^{(2)}_6 = C'^{(2)} x'^{(2)}_6$
$\varnothing$	$\varnothing$

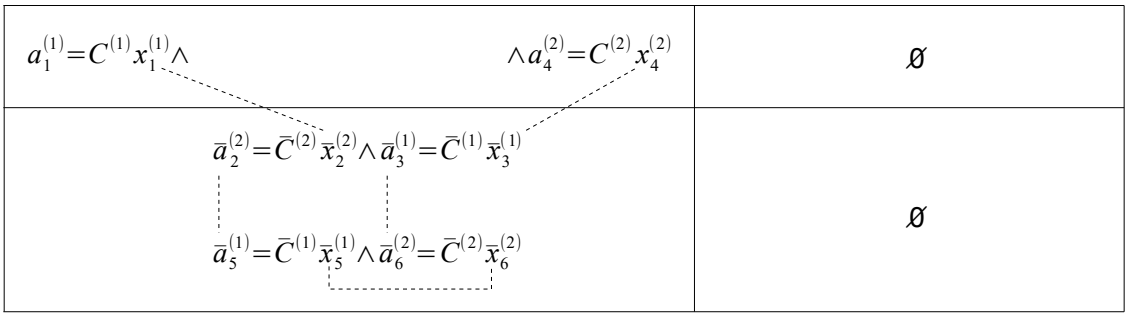
На таких гребнях определим конструктивную операцию M_b , которая переводит в нижние клетки соединённые пунктиром равенства из разных клеток с соответствующей заменой сортов:



Эта операция является одновременным применением операций

$$M_b^{(i)}x^{(i)}\to\bar{x}^{(i)},M_b^{(i)}x'^{(i)}\to\bar{x}'^{(i)},M_b^{(i)}a^{(i)}\to\bar{a}^{(i)},M_b^{(i)}a'^{(i)}\to\bar{a}'^{(i)},i=1,2.$$

Далее, определим конструктивную операцию M_l, которая переводит с соответствующей заменой сортов из правых клеток в левые клетки соединённые пунктиром равенства:



Эти операции яляются одновременным применением наборов операций

$$M_l^{(i)}x'^{(i)}\to x^{(i)},M_l^{(i)}\bar{x}'^{(i)}\to\bar{x}^{(i)},M_l^{(i)}a'^{(i)}\to a^{(i)},M_l^{(i)}\bar{a}'^{(i)}\to\bar{a}^{(i)}.$$

Введём также операции, противоположные предыдущим

$$M_r^{(i)}x^{(i)}\to x'^{(i)},M_r^{(i)}\bar{x}^{(i)}\to\bar{x}'^{(i)},M_r^{(i)}a^{(i)}\to a'^{(i)},M_r^{(i)}\bar{a}^{(i)}\to\bar{a}'^{(i)}.$$

Для направленных бинарных склеек вводится тернарная операция умножения μ, согласованная с проекциями с помощью аксиом

$$C^{(1)}\mu^{(1)}x_p^{(1)}x_q^{(1)}x_r^{(1)}=C^{(1)}x_p^{(1)},C^{(2)}\mu^{(2)}x_p^{(2)}x_q^{(2)}x_r^{(2)}=C^{(2)}x_q^{(2)},\qquad(6)$$

где $x_p^{(1)},x_q^{(1)},x_p^{(2)},x_q^{(2)}$ - переменные перемножаемых первых двух связных гребней и $x_r^{(2)},x_r^{(2)}$ - переменные третьего из перемножаемых трех гребней соответствующих свёртке с переменными $x'^{(2)}_r x'^{(2)}_r$.

Данные аксиомы задают конструктивные операции ***согласования умножения с проекциями***, применяемыми по отдельности следующими правилами подстановки:

$$C^{(1)}\mu^{(1)}x_p^{(1)}x_q^{(1)}x_r^{(1)} \to C^{(1)}x_p^{(1)}, C^{(1)}x_p^{(1)} \to C^{(1)}\mu^{(1)}x_p^{(1)}x_q^{(1)}x_r^{(1)},$$
$$C^{(2)}\mu^{(2)}x_p^{(2)}x_q^{(2)}x_r^{(2)} \to C^{(2)}x_q^{(2)}, C^{(2)}x_q^{(2)} \to C^{(2)}\mu^{(2)}x_p^{(2)}x_q^{(2)}x_r^{(2)}.$$

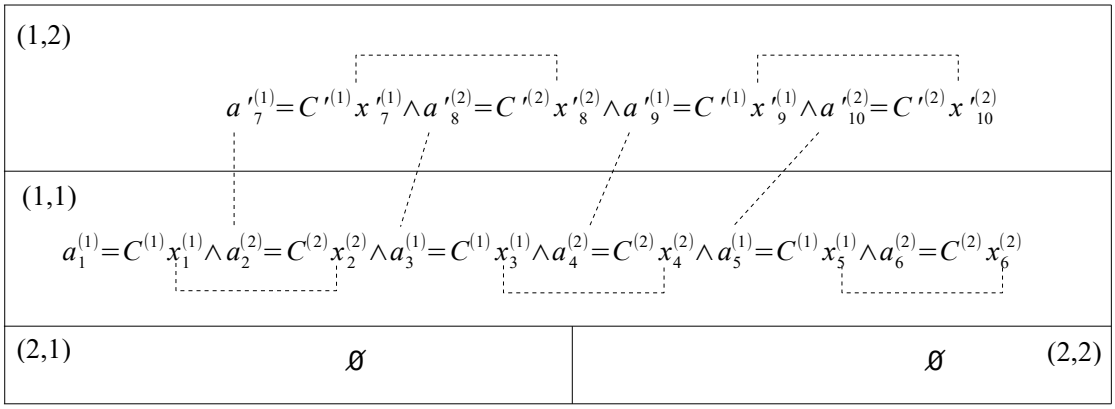
Данные согласования отвечают перемножению гребней, при котором не меняются начальная проекция у первого сомножителя и конечная проекция у второго сомножителя, так же, как это имеет место при перемножении стрелок в категориях. Далее, определяется с помощью пятиместной операции умножения случай, когда указанные проекции также изменяются при перемножении гребней.

Помимо конструктивных операций M_b, M_l для полного определения умножения гребней вводится конструктивная операция *Ev*, переводящая предыдущую формулу в следующую формулу

$a_1^{(1)}=C^{(1)}\mu^{(1)}x_1^{(1)}x_3^{(1)}x_5^{(1)}\wedge \qquad \wedge a_4^{(2)}=C^{(2)}\mu^{(2)}x_2^{(2)}x_4^{(2)}x_6^{(2)}$	$\varnothing$
$\bar{a}_2^{(2)}=\bar{C}^{(2)}\bar{x}_2^{(2)}\wedge\bar{a}_3^{(1)}=\bar{C}^{(1)}\bar{x}_3^{(1)}$ $\bar{a}_5^{(1)}=\bar{C}^{(1)}\bar{x}_5^{(1)}\wedge\bar{a}_6^{(2)}=\bar{C}^{(2)}\bar{x}_6^{(2)}$	$\varnothing$

Полученный полный гребень даёт произведение двух базовых гребней с помощью имеющейся свёртки, то есть задает бинарную операцию, обозначим ее через λ_2 (для двух внешних гребней, $\lambda_2 x_1 x_3 x'_5$, а свёртка x'_5 является параметром), и является результатом применения соответствующих конструктивных операций. Умножение пары уже построенных гребней по определению проводится также, при этом очевидным образом накапливаются внутренние гребни. Полный гребень содержит в виде своего внутреннего гребня историю получения из базовых гребней. Для теории систем этот механизм формализует получение системы из других систем, становящихся её подсистемами [7].

Рассмотрим следующий полный гребень (клетка (1,2) расположена над клеткой (1,1)).



Результат применения имеющихся свёрток зависит от порядка применения свёрток. Если сначала применяется левая свёртка, то получаем

$a_1^{(1)} = C^{(1)} \mu^{(1)} \mu^{(1)} x_1^{(1)} x_3^{(1)} x_7^{(1)} x_5^{(1)} x_9^{(1)} \wedge a_6^{(2)} = C^{(2)} \mu^{(2)} \mu^{(2)} x_2^{(2)} x_4^{(2)} x_8^{(2)} x_6^{(2)} x_{10}^{(2)}$	∅
$\bar{a}_2^{(2)} = \bar{C}^{(2)} \bar{x}_2^{(2)} \wedge \bar{a}_3^{(1)} = \bar{C}^{(1)} \bar{x}_3^{(1)} \wedge \bar{a}_7^{(1)} = \bar{C}^{(1)} \bar{x}_7^{(1)} \wedge \bar{a}_8^{(2)} = \bar{C}^{(2)} \bar{x}_8^{(2)}$ $\bar{a}_4^{(2)} = \bar{C}^{(2)} \bar{\mu}^{(2)} \bar{x}_2^{(2)} \bar{x}_4^{(2)} \bar{x}_8^{(2)} \wedge \bar{a}_5^{(1)} = \bar{C}^{(1)} \bar{x}_5^{(1)} \wedge \bar{a}_9^{(1)} = \bar{C}^{(1)} \bar{x}_9^{(1)} \wedge \bar{a}_{10}^{(2)} = \bar{C}^{(2)} \bar{x}_{10}^{(2)}$	∅

Если сначала применяется правая свёртка, то получаем

$a_1^{(1)} = C^{(1)} \mu^{(1)} x_1^{(1)} \mu^{(1)} x_3^{(1)} x_5^{(1)} x_9^{(1)} x_7^{(1)} \wedge a_6^{(2)} = C^{(2)} \mu^{(2)} x_2^{(2)} \mu^{(2)} x_4^{(2)} x_6^{(2)} x_{10}^{(2)} x_8^{(2)}$	∅
$\bar{a}_2^{(2)} = \bar{C}^{(2)} \bar{x}_2^{(2)} \wedge \bar{a}_3^{(1)} = \bar{C}^{(1)} \bar{\mu}^{(1)} \bar{x}_3^{(1)} \bar{x}_5^{(1)} \bar{x}_9^{(1)} \wedge \bar{a}_7^{(1)} = \bar{C}^{(1)} \bar{x}_7^{(1)} \wedge \bar{a}_8^{(2)} = \bar{C}^{(2)} \bar{x}_8^{(2)}$ $\bar{a}_4^{(2)} = \bar{C}^{(2)} \bar{x}_4^{(2)} \wedge \bar{a}_5^{(1)} = \bar{C}^{(1)} \bar{x}_5^{(1)} \wedge \bar{a}_9^{(1)} = \bar{C}^{(1)} \bar{x}_9^{(1)} \wedge \bar{a}_{10}^{(2)} = \bar{C}^{(2)} \bar{x}_{10}^{(2)}$	∅

Определение

Формулу

$$\mu^{(i)}u^{(i)}\mu^{(i)}v^{(i)}w^{(i)}r^{(i)}s^{(i)}=\mu^{(i)}\mu^{(i)}u^{(i)}v^{(i)}r^{(i)}w^{(i)}s^{(i)},\ i=1,2 \qquad (7)$$

назовём ***аксиомой ассоциативности***, она определяет конструктивные операции с помощью следующих действующих в соответствующих алгорифмах по отдельности формул подстановки

$$\mu^{(i)}u^{(i)}\mu^{(i)}v^{(i)}w^{(i)}r^{(i)}s^{(i)}\to\mu^{(i)}\mu^{(i)}u^{(i)}v^{(i)}r^{(i)}w^{(i)}s^{(i)},$$
$$\mu^{(i)}\mu^{(i)}u^{(i)}v^{(i)}r^{(i)}w^{(i)}s^{(i)}\to\mu^{(i)}u^{(i)}\mu^{(i)}v^{(i)}w^{(i)}r^{(i)}s^{(i)}.$$

Для бинарной операции на гребнях имеем (сорта не указаны)

$$\lambda_2 u\lambda_2 v w r' s'\to\lambda_2\lambda_2 u v r' w s',$$
$$\lambda_2\lambda_2 u v r' w s'\to\lambda_2 u\lambda_2 v w r' s'.$$

Если имеется гребень с подформулой $a^{(i)}=C^{(i)}\mu^{(i)}u^{(i)}\mu^{(i)}v^{(i)}w^{(i)}r^{(i)}s^{(i)}$, то используя аксиому ассоциативности можно получить (например, если внутренний гребень у исходного гребня пустой) в качестве гребня новый гребень с подформулой $a^{(i)}=C^{(1)}\mu^{(i)}\mu^{(i)}u^{(i)}v^{(i)}r^{(i)}w^{(i)}s^{(i)}$. Этот новый гребень вполне может быть получен другим способом (с помощью операций свёртки и проекций). Подчеркнём, что мы не заботимся о том, чтобы использовать некоторый *минимальный* набор конструктивных операций для построения направленных бинарных склеек.

Далее мы будем операцию μ обозначать μ_t.

3. Категорные диаграммы

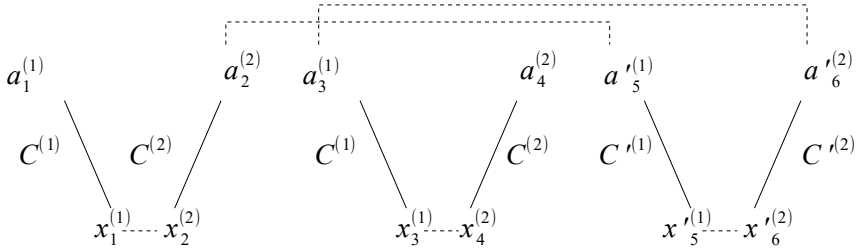
Воспользуемся приёмом, который использован во введенной в [6] аксиоматике теории категорий для графической записи формул языка. Неформальное обсуждение и детали аналога этого применения для категорных склеек можно найти в [7].

Определим графическое изображение формул.

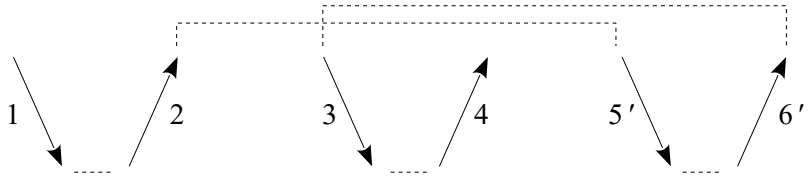
Основной полный гребень со свёрткой

$a_1^{(1)}=C^{(1)}x_1^{(1)}\wedge a_2^{(2)}=C^{(2)}x_2^{(2)}\wedge a_3^{(1)}=C^{(1)}x_3^{(1)}\wedge a_4^{(2)}=C^{(2)}x_4^{(2)}$	$a'^{(1)}_5=C'^{(1)}x'^{(1)}_5\wedge a'^{(2)}_6=C'^{(2)}x'^{(2)}_6$
$\varnothing$	$\varnothing$

имеет изображение



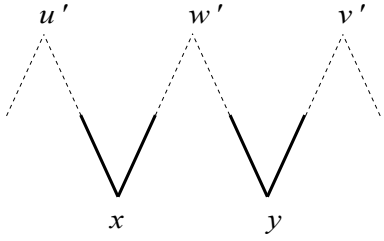
или в кратком варианте



(8)

4. Общая бинарная операция в бинарных направленных склейках

Рассмотрим следующую бинарную операцию λ на гребнях $z=\lambda u'xw'yv'$, (u', w', v' рассмтриваются как параметры бинарной операции λ от двух внешних гребней x и y), изменяющую все проекции сомножителей и выполняемую аналогичным приведённым выше вариантам

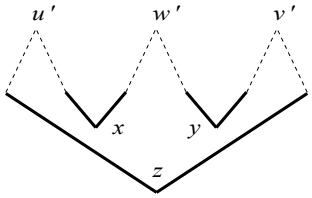


После выполнения операций M (с индексами) и Ev получаем внешний гребень z

$$C^{(1)}z^{(1)}=C^{(1)}u^{(1)}\wedge C^{(2)}z^{(2)}=C^{(2)}v^{(2)}\wedge z^{(1)}=z^{(2)} \quad ,$$
$$z^{(i)}=\lambda^{(i)}u'^{(i)}x^{(i)}w'^{(i)}y^{(i)}v'^{(i)};$$
$$\lambda^{(i)}u'^{(i)}x^{(i)}w'^{(i)}y^{(i)}v'^{(i)}=\mu_5^{(i)}u^{(i)}x^{(i)}w^{(i)}y^{(i)}v^{(i)}, i=1,2 \quad .$$

Дополняя этим гребнем предыдущий гребень, получим

(9)



Операция μ_5 на пяти гребнях определяет также тернарную операцию ρ на гребнях, которая нам понадобится

$$\rho^{(i)}u^{(i)}x'^{(i)}w^{(i)}y'^{(i)}v^{(i)}=\mu_5^{(i)}u^{(i)}x^{(i)}w^{(i)}y^{(i)}v^{(i)}, i=1,2$$

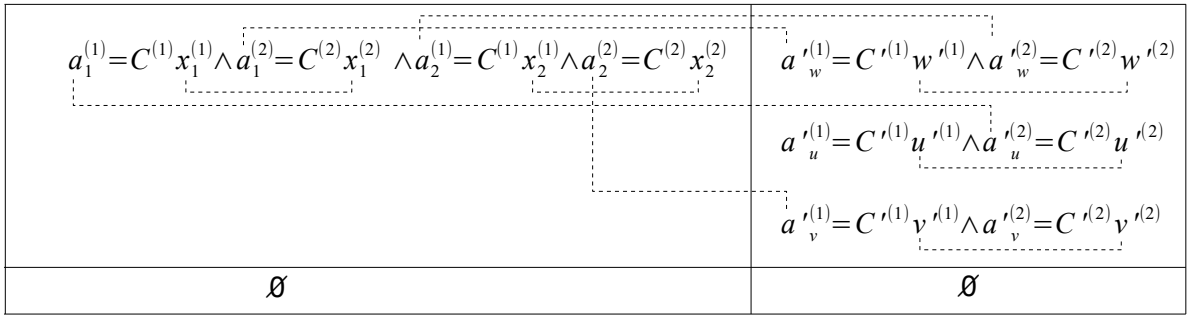
поскольку $x^{(1)}=\xi_{13}x'^{(1)}, x^{(2)}=\xi_{24}x'^{(2)}, y^{(1)}=\xi_{13}y'^{(1)}, y^{(2)}=\xi_{24}y'^{(2)},$ то вводим операцию «снятия штриха» $\xi^-=(\xi_{13},\xi_{24}), \xi^-(x')=\xi^-(x'^{(1)},x'^{(2)})=(x^{(1)},x^{(2)})$ имеем

$$\rho^{(i)}u^{(i)}x'^{(i)}w^{(i)}y'^{(i)}v^{(i)}=\mu_5^{(i)}u^{(i)}\xi^-x'^{(i)}w^{(i)}\xi^-y'^{(i)}v^{(i)}=$$
$$=\mu_5^{(i)}\xi_{II}^-\xi_{IV}^-u^{(i)}x'^{(i)}w^{(i)}y'^{(i)}v^{(i)}, \rho^{(i)}=\mu_5^{(i)}\xi_{II}^-\xi_{IV}^-$$

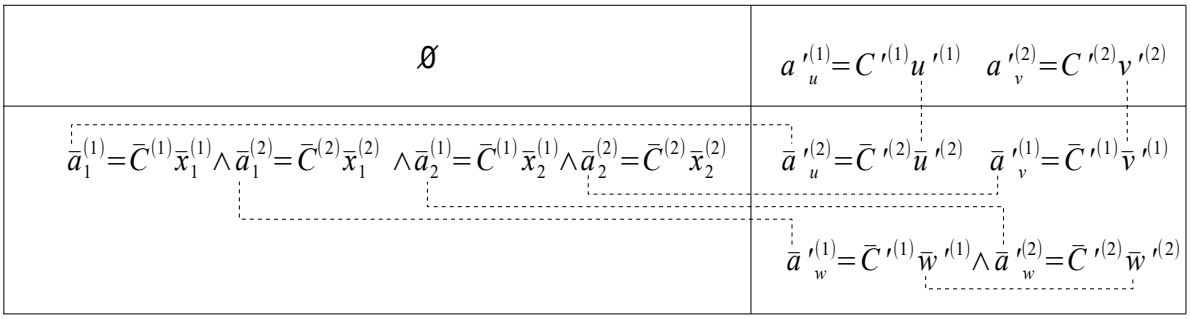
Аналогично имеем $\lambda^{(i)}=\mu_5^{(i)}\xi_I^-\xi_{III}^-\xi_V^-$.

Приведём конструктивное описание операции λ (и μ_5).

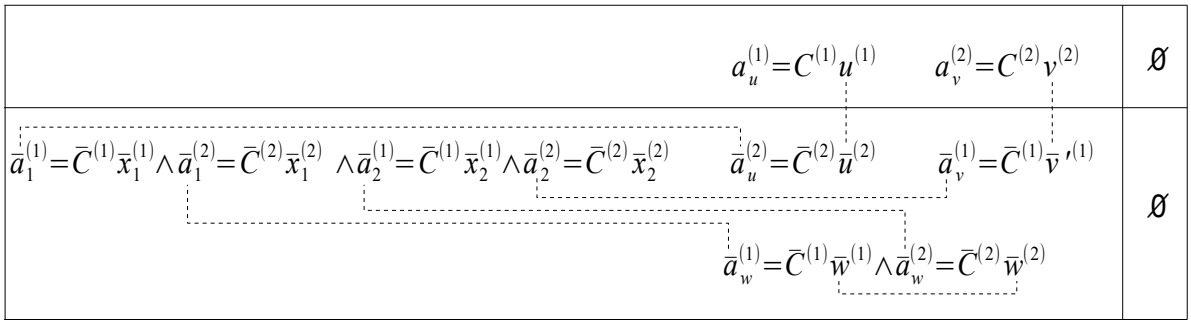
Полный гребень имеет вид



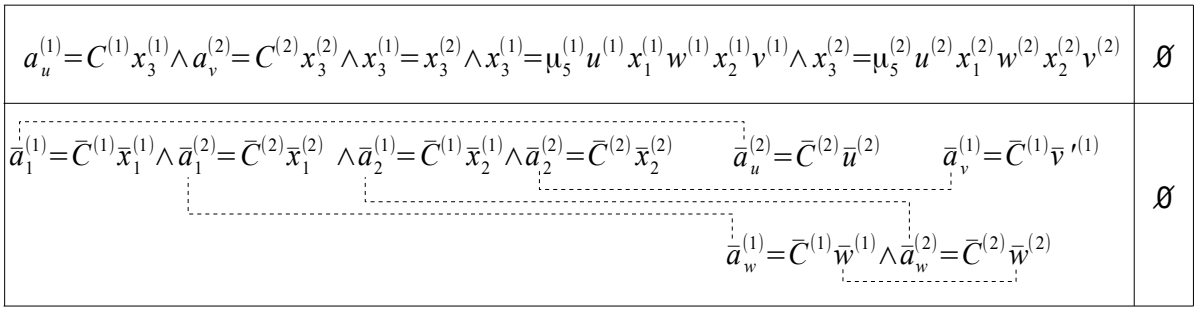
Применим операцию M_b , получим



Применим операцию M_l , получим

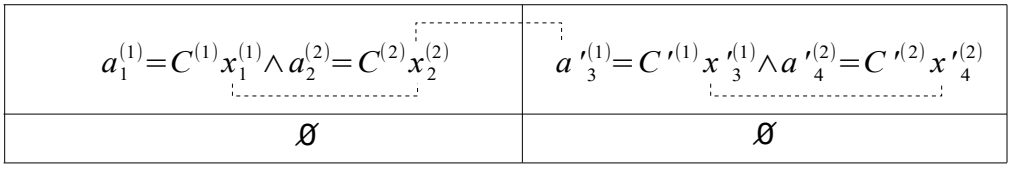


Применим операцию *Ev*, получим окончательно, вводя обозначение для требуемой пятиместной опреации μ_5 на пяти внешних гребнях *u, x*$_1$*, w, x*$_2$*, v*



Частным случаем операции λ является следующее построение её варианта из подходящих бинарных операций.

Определим следующую бинарную (по гребню и свёртке) операцию μ_r , действующую на полный гребень



Применяем операцию M_b , получаем

$a_1^{(1)}=C^{(1)}x_1^{(1)}\wedge$	$\wedge a'^{(2)}_4=C'^{(2)}x'^{(2)}_4$
$\bar{a}_2^{(2)}=\bar{C}^{(2)}\bar{x}_2^{(2)}$	$\bar{a}'^{(1)}_3=\bar{C}'^{(1)}\bar{x}'^{(1)}_3$

Применяем операцию M_l , получаем

$a_1^{(1)}=C^{(1)}x_1^{(1)}\wedge \quad \wedge a_4^{(2)}=C^{(2)}x_4^{(2)}$	$\varnothing$
$\bar{a}_2^{(2)}=\bar{C}^{(2)}\bar{x}_2^{(2)} \quad \bar{a}_3^{(1)}=\bar{C}^{(1)}\bar{x}_3^{(1)}$	$\varnothing$

Применяем операцию *Ev*, получаем

$a_1^{(1)}=C^{(1)}\mu_r^{(1)}x_1^{(1)}x_3^{(1)}\wedge a_4^{(2)}=C^{(2)}\mu_r^{(2)}x_2^{(2)}x_4^{(2)}$	$\varnothing$
$\bar{a}_2^{(2)}=\bar{C}^{(2)}\bar{x}_2^{(2)} \quad \bar{a}_3^{(1)}=\bar{C}^{(1)}\bar{x}_3^{(1)}$	$\varnothing$

Необходимую бинарную операцию на двух внешних гребнях мы обозначили той же буквой μ_r.

Определим следующую бинарную (по гребню и свёртке) операцию μ_l, действующую на полный гребень

$a_1^{(1)}=C^{(1)}x_1^{(1)}\wedge a_2^{(2)}=C^{(2)}x_2^{(2)}$	$a'^{(1)}_3=C'^{(1)}x'^{(1)}_3\wedge a'^{(2)}_4=C'^{(2)}x'^{(2)}_4$
$\varnothing$	$\varnothing$

Применяем операции M_b , M_l , *Ev*, получаем

$a_1^{(1)}=C^{(1)}\mu_l^{(1)}x_3^{(1)}x_1^{(1)}\wedge a_4^{(2)}=C^{(2)}\mu_l^{(2)}x_4^{(2)}x_2^{(2)}$	$\varnothing$
$\bar{a}_4^{(2)}=\bar{C}^{(2)}\bar{x}_4^{(2)}\qquad\bar{a}_1^{(1)}=\bar{C}^{(1)}\bar{x}_1^{(1)}$	$\varnothing$

Необходимую бинарную операцию на двух внешних гребнях мы обозначили той же буквой μ_l.

Операции μ_r, μ_l, также как и μ_t, согласованы с проекциями.

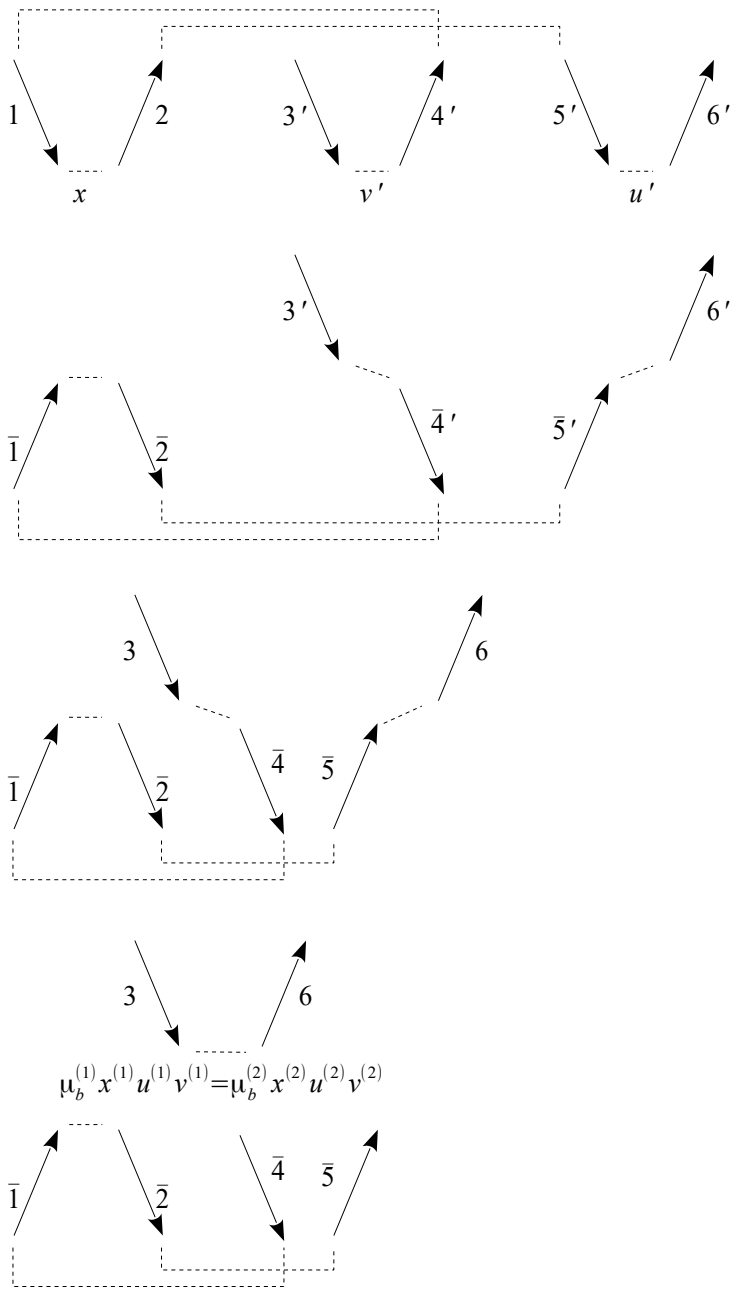
Тернарную операцию $\mu_t^{(i)}x^{(i)}y^{(i)}z^{(i)}$ можно сузить до операции (если для определённости принять, что сначала действует операция μ_l и затем μ_r)

$$\mu_t^{(i)}x^{(i)}y^{(i)}z^{(i)}\to\mu_r^{(i)}x^{(i)}\mu_l^{(i)}y^{(i)}z^{(i)},i=1,2.$$

Введём ещё тернарную (по гребню и двум свёрткам) операцию μ_b, следующим образом, используя аналогичным способом имеющиеся операции.

Сводящийся к бинарным операциям вариант этой операции имеет вид (индекс сорта не приводится, для определённости сначала примем действие операции μ_r, потом операции μ_l)

$$\mu_b x\,u\,v\stackrel{\text{def}}{=}\mu_l\mu_r x\,u\,v$$



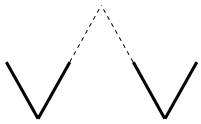
Теперь мы можем получить из бинарных операций вариант общей бинарной операции направленных бинарных склеек, при этом индекс 5 можно уже не использовать (индекс сорта не приводится):

$$\mu u\, x\, w\, y\, v \stackrel{\text{def}}{=} \mu_b \mu_t x\, y\, w\, u\, v \stackrel{\text{def}}{=} \mu_l \mu_r \mu_r x \mu_l\, y\, w\, u\, v \quad .$$

На переменные *u, w, v* можно смотреть как на параметры для операции перемножения двух гребней *x* и *y*.

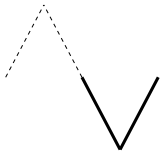
5. Нерв бинарной направленной склейки

Рассмотрим конструктивную операцию выписывания следующего вида формул. Первый шаг - это операция P_0 построения формулы (8), которую мы изобразим в сокращённом виде (здесь свёртки выписываем вверху пунктиром)



Назовем эту формулу ***цепочкой гребней*** из двух звеньев.

Следующей конструктивной операцией P определим приписывание справа к имещейся цепочке формулы



с результатом в виде диаграммы ***цепочки гребней*** из трех звеньев

Эту формулу можно двумя разными способами, выполняя в той или иной последовательности имеющиеся свёртки (здесь используется операция μ_t) превратить в соответствующий полный гребень. Поэтому ***цепочкой из трех звеньев*** назовём два объекта: саму диаграмму и порядок возможного перемножения гребней. Если три гребня обозначить соответственно *x, y, z*, то мы получим две цепочки: $\mu_t x \mu_t y z$ и $\mu_t \mu_t x y z$.

Заменяя μ_t на операцию λ мы получим общий случай цепочек направленной склейки, который рассмотрим детальнее далее.

Определение

Формула, полученная применением указанных операций с указанием порядка применения свёрток называется ***цепочкой гребней***. Набор трёх указанных конструктивных операций (P_0, P и указание порядка) называется ***конструктивным нервом*** данной категорной склейки.

Полезно определить бинарную на цепочках ***операцию соединения цепочек,*** то есть операцию приписывания справа гребня к имеющемуся гребню, для чего достаточно приписывания свёртки (или бинарной операции μ_t, λ и др.), соединяющей проекцию второго

сорта последнего гребня с проекцией первого сорта первого гребня присоединяемой слева цепочки, и определения порядка: порядок первой цепочки, порядок второй цепочки и затем применение добавленной свёртки или операции.

Мы оставляем те же названия для цепочек, в которых вместо указанных свёрток используется операция λ.

Перейдём к конструктивным операциям дуальности.

6. Сортовая дуальность

Этот вид дуальности определяет общеизвестную дуальность в категориях, ряд основных понятий теории категорий, таких как понятия дуальной или двойственной категории, ковариантного и контравариантного функторов и других связанных с двойственностью категорных понятий.

Описание сортовой дуальности состоит из нескольких шагов.

Вводится другой алфавит, взаимно однозначно соответствующий алфавиту имеющейся склейки. Далее, с помощью тех же конструктивных операций в новом алфавите строится язык для другой формальной склейки, далее, устанавливаются отвечающие сортовой дуальности связи с первоначальной склейкой, позволяющие все операции в новой склейке получить из операций исходной. Указанные шаги удобно проводить, расширив, как указано, начальный алфавит его копией, использовав переобозначения букв, и построив формальную склейку в этом новом алфавите. В расширенном языке мы получаем две формальные склейки, взаимоотношения которых (сортовую дуальность) можно оформить подходящими функциональными буквами. Реализуем эту программу, уточняя необходимые детали.

Расширим построенный язык, добавив его копию, для чего вводим:

- восемь дополнительных сортов $s_1 = \{ s_1^{(i)}, s_1{}'^{(i)}, \bar{s}'^{(i)}_1, \bar{s}^{(i)}_1 \}$, $i = 1,2$ для переменных (предыдущие сорта s_0 обозначим теми же буквами, заменив индекс 1 на 0, то есть использовав обозначения $s^{(\,)}{}_0$);

- новые буквы ($i = 1,2$, $\alpha, \beta = 1, \ldots, 8$, k, m - натуральные числа, знак равенства оставим тем же, как и знак конъюнкции)

$$y_k^{(i)}, b_m^{(i)}, y'^{(i)}_k, b'^{(i)}_m, \bar{y}'^{(i)}_k, \bar{b}'^{(i)}_m, \bar{y}^{(i)}_k, \bar{b}^{(i)}_m, G^{(i)}, G'^{(i)}, \bar{G}^{(i)}, \bar{G}'^{(i)}, \nu^{(i)}, \theta^{(i)}, \sigma^{(i)}, =_{\alpha\beta}, \eta_{\alpha\beta}, \wedge \ ,$$

соответствующие имеющимся буквам

$$x_k^{(i)}, a_m^{(i)}, x'^{(i)}_k, a'^{(i)}_m, \bar{x}'^{(i)}_k, \bar{a}'^{(i)}_m, \bar{x}^{(i)}_k, \bar{a}^{(i)}_m, C^{(i)}, C'^{(i)}, \bar{C}^{(i)}, \bar{C}'^{(i)}, \mu^{(i)}, \lambda^{(i)}, \rho^{(i)}, =_{\alpha\beta}, \xi_{\alpha\beta}, \wedge \ .$$

Мы имеем конструктивную операцию замены формул и выражений, порожденную сменой алфавита, обозначим её R («*replacement*», переводит исходный язык в язык с другим

равномощным алфавитом), а также имеем обратную для нее R^{-1}(или короче R^{-}). Введённую копию языка будем называть ***R-копией***. Итак, для рассмотрения дуальности мы изучаем ***объединенный язык***, объединение языка склеек и его *R*-копии.

Для рассмотрения дуальности введём набор D функциональных символов и набор D^{-} обратных к ним символов (тип указан в скобках, $i \neq j$, i, j =1,2,)

$$D^{(i)}\left(s_0^{(i)} \to s_1^{(j)}\right), D'^{(i)}\left(s_0'^{(i)} \to s_1'^{(j)}\right), \bar{D}'^{(i)}\left(\bar{s}'^{(i)}_0 \to \bar{s}'^{(j)}_1\right), \bar{D}^{(i)}\left(\bar{s}^{(i)}_0, \to \bar{s}^{(j)}_1\right),$$
$$D^{-(i)}\left(s_1^{(i)} \to s_0^{(j)}\right), D'^{-(i)}\left(s_1'^{(i)} \to s_0'^{(j)}\right), \bar{D}'^{-(i)}\left(\bar{s}'^{(i)}_1 \to \bar{s}'^{(j)}_0\right), \bar{D}^{-(i)}\left(\bar{s}^{(i)}_1, \to \bar{s}^{(j)}_0\right);$$
$$Dx_k^{(i)} \to y_k^{(j)}, Da_k^{(i)} \to b_k^{(j)}, Dx'^{(i)}_k \to y'^{(j)}_k, Da'^{(i)}_k \to b'^{(j)}_k,$$
$$D\bar{x}'^{(i)}_k \to \bar{y}'^{(j)}_k, D\bar{a}'^{(i)}_k \to \bar{b}'^{(i)}_j, D\bar{x}_k^{(i)} \to \bar{y}_k^{(i)}, D\bar{a}_k^{(i)} \to \bar{b}_j^{(i)},$$
$$D^{-} y_k^{(i)} \to x_k^{(j)}, D^{-} b_k^{(i)} \to a_k^{(j)}, D^{-} y'^{(i)}_k \to x'^{(j)}_k, D^{-} b'^{(i)}_k \to a'^{(j)}_k,$$
$$D^{-} \bar{y}'^{(i)}_k \to \bar{x}'^{(j)}_k, D^{-} \bar{b}'^{(i)}_k \to \bar{a}'^{(i)}_j, D^{-} \bar{y}_k^{(i)} \to \bar{x}_k^{(i)}, D^{-} \bar{b}_k^{(i)} \to \bar{a}_j^{(i)}.$$

Определим действие проекций G через проекции C.

$$G^{(i)} y_k^{(i)} \stackrel{\text{def}}{=} DC^{(j)} D^{-} y_k^{(i)}, G'^{(i)} y'^{(i)}_k \stackrel{\text{def}}{=} DC'^{(j)} D^{-} y'^{(i)}_k,$$
$$\bar{G}^{(i)} \bar{y}_k^{(i)} \stackrel{\text{def}}{=} D\bar{C}^{(j)} D^{-} \bar{y}_k^{(i)}, \bar{G}'^{(i)} \bar{y}'^{(i)}_k \stackrel{\text{def}}{=} D\bar{C}'^{(j)} D^{-} \bar{y}'^{(i)}_k.$$

Определим аналогичную связь операций D, ξ_{ij}, η_{ij} соответствиями (здесь i, j берутся из набора чисел от 1 до 8) $\eta_{i-1\,j-1} \stackrel{\text{def}}{=} D\xi_{i\,j} D^{-}$, *i, j* - чётные числа; $\eta_{i+1\,j+1} = D\xi_{i\,j} D^{-}$, *i, j* - нечётные числа; $\eta_{i-1\,j+1} = D\xi_{i\,j} D^{-}$, *i,* - чётное число, *j* - нечётное число; $\eta_{i+1\,j-1} = D\xi_{i\,j} D^{-}$ *i*, - нечётное число, *j* - чётное число.

Дополняем конструктивные операции D и R до операций $\hat{D}$ и $\hat{R}$, включающих преобразования $\hat{D}: t_1 = t_2 \to Dt_1 = Dt_2, \hat{R}: t_1 = t_2 \to Rt_1 = Rt_2$ для термов s_0 сортов и $\hat{D}: A \wedge B \to \hat{D}A \wedge \hat{D}A, \hat{R}: A \wedge B \to \hat{R}A \wedge \hat{R}A$ для формул с конъюнкциями.

Определим действие операции θ через операцию λ. Для этого цепочку из двух перемножаемых гребней переводим операцией $\hat{D}$ в цепочку из сортов s_1 и применяем операцию θ.

$$a'^{(1)}_u = C'^{(1)} u'^{(1)} \wedge a'^{(2)}_u = C'^{(2)} u'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge a'^{(2)}_u = a_1^{(1)} \wedge$$
$$\wedge a_1^{(1)} = C^{(1)} x_1^{(1)} \wedge a_1^{(2)} = C^{(2)} x_1^{(2)} \wedge x_1^{(1)} = x_1^{(2)} \wedge a_1^{(2)} = a'^{(1)}_w \wedge$$
$$\wedge a'^{(1)}_w = C'^{(1)} w'^{(1)} \wedge a'^{(2)}_w = C'^{(2)} w'^{(2)} \wedge w'^{(1)} = w'^{(2)} \wedge a'^{(2)}_w = a_2^{(1)} \wedge$$
$$\wedge a_2^{(1)} = C^{(1)} x_2^{(1)} \wedge a_2^{(2)} = C^{(2)} x_2^{(2)} \wedge x_2^{(1)} = x_2^{(2)} \wedge a_2^{(2)} = a'^{(1)}_v \wedge$$
$$\wedge a'^{(1)}_v = C'^{(1)} v'^{(1)} \wedge a'^{(2)}_v = C'^{(2)} v'^{(2)} \wedge v'^{(1)} = v'^{(2)}.$$

Применяем операцию $\hat{D}$, имеем

$$b'^{(2)}_{p}=G'^{(2)}p'^{(2)}\wedge b'^{(1)}_{p}=G'^{(1)}p'^{(1)}\wedge p'^{(2)}=p'^{(1)}\wedge b'^{(1)}_{p}=b_1^{(2)}\wedge$$
$$\wedge b_1^{(2)}=G^{(2)}y_1^{(2)}\wedge b_1^{(1)}=G^{(1)}y_1^{(1)}\wedge y_1^{(2)}=y_1^{(1)}\wedge b_1^{(1)}=b'^{(2)}_{q}\wedge$$
$$\wedge b'^{(2)}_{q}=G'^{(2)}q'^{(2)}\wedge b'^{(1)}_{q}=G'^{(1)}q'^{(1)}\wedge q'^{(2)}=q'^{(1)}\wedge b'^{(1)}_{q}=b_2^{(2)}\wedge$$
$$\wedge b_2^{(2)}=G^{(2)}y_2^{(2)}\wedge b_2^{(1)}=G^{(1)}y_2^{(1)}\wedge y_2^{(2)}=y_2^{(1)}\wedge b_2^{(1)}=b'^{(2)}_{r}\wedge$$
$$\wedge b'^{(2)}_{r}=G'^{(2)}r'^{(2)}\wedge b'^{(1)}_{r}=G'^{(1)}r'^{(1)}\wedge r'^{(2)}=r'^{(1)}.$$

Пользуясь коммутативностью и ассоциативностью конъюнкции переместим равенства в тот порядок, который требуется для применения операции θ.

$$b'^{(1)}_{r}=G'^{(1)}r'^{(1)}\wedge b'^{(2)}_{r}=G'^{(2)}r'^{(2)}\wedge r'^{(1)}=r'^{(2)}$$
$$\wedge b'^{(2)}_{r}=b_2^{(1)}\wedge b_2^{(1)}=G^{(1)}y_2^{(1)}\wedge b_2^{(2)}=G^{(2)}y_2^{(2)}\wedge y_2^{(1)}=y_2^{(2)}\wedge$$
$$\wedge b_2^{(2)}=b'^{(1)}_{q}\wedge b'^{(1)}_{q}=G'^{(1)}q'^{(1)}\wedge b'^{(2)}_{q}=G'^{(2)}q'^{(2)}\wedge q'^{(1)}=q'^{(2)}\wedge$$
$$\wedge b'^{(2)}_{q}=b_1^{(1)}\wedge b_1^{(1)}=G^{(1)}y_1^{(1)}\wedge b_1^{(2)}=G^{(2)}y_1^{(2)}\wedge y_1^{(1)}=y_1^{(2)}\wedge$$
$$\wedge b_1^{(2)}=b'^{(1)}_{p}\wedge b'^{(1)}_{p}=G'^{(1)}p'^{(1)}\wedge b'^{(2)}_{p}=G'^{(2)}p'^{(2)}\wedge p'^{(1)}=p'^{(2)}.$$

Итак, имеем цепочку $\theta r'y_2q'y_1p'$, полученную из цепочки $\lambda u'x_1w'x_2v'$ или окончательно (при условиях $Du^{(i)}\to p^{(j)}, Dw^{(i)}\to q^{(j)}, Dv^{(i)}\to r^{(j)}, i\neq j, i, j=1,2$) имеем

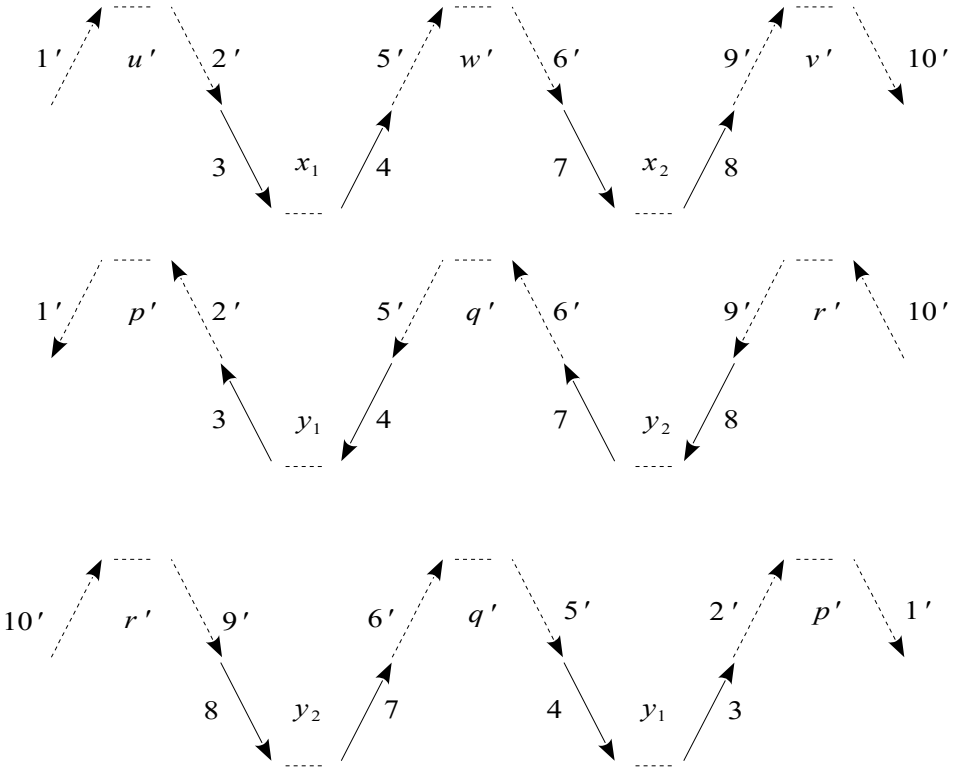
$$\theta r'y_1q'y_2p'=D\lambda u'x_2w'x_1v'.$$

Для операции σ отсюда имеем выражение через операцию ρ

$$\sigma r y'_1 q y'_2 p=D\rho u x'_2 w x'_1 v.$$

Для операции ν имеем выражение через μ операцию $\nu r y_1 q y_2 p=D\mu u x_2 w x_1 v$.

Приведём наглядную схему выкладок для вычисления θ через λ.



Из первой схемы вторая получается операцией *D*, примененной к каждому элементу первой схемы, третья схема является приведением записи второй схемы для возможного проведения конструктивной операции умножения ν .

Определение

Формулы, полученные с помощью набора операций $\hat{D}$, назовем ***дуальными по сортам*** к исходным формулам. Возникающую при действии $\hat{D}$ категорную склейку назовём ***дуальной по сортам*** к исходной направленной бинарной склейке.

Частным случаем приведённого построения является формализация примера контравариантного функтора в теории категорий. Накладывая те или иные условия на функциональные буквы, аналогичные *D*, мы получаем формализацию понятия функтора для категорных склеек (здесь направленных бинарных склеек).

Таким образом, мы задали конструктивные операции перехода от категорной склейки к дуальной по сортам категорной склейке (в нашем случае для направленных бинарных склеек).

7. Свёрточная дуальность

Этот вид дуальности открыт автором и оказался полезным в теории мультикатегорий, свёрточный вариант которых адекватно моделирует искусственные нейронные сети произвольной топологии [7, 8].

Введём, как и в предыдущем разделе, копию в тех же обозначениях формального языка для дуальной по свёрткам склейки. Как и ранее определена конструктивная операция R и ее обратная R^-, переводящая символы и выражения исходного языка в его копию и наоборот.

Построим для заданной склейки дуальную по свёрткам склейку, добавив следующий набор $B=\{B^0, B', \bar{B}, \bar{B}', \dots\}$ конструктивных операций (и набор B^- им обратных), применяемых после построения гребня или свёртки исходной склейки (используемую копию исходной склейки назовём *B*-копией):

$$B^0: a_k^{(i)} \to b'^{(i)}_k;\ B^0: x_k^{(i)} \to y'^{(i)}_k; \qquad B': a'^{(i)}_k \to b_k^{(i)};\ B': x'^{(i)}_k \to y_k^{(i)} \ ;$$

$$\bar{B}: \bar{a}_k^{(i)} \to \bar{b}'^{(i)}_k;\ \bar{B}: \bar{x}_k^{(i)} \to \bar{y}'^{(i)}_k \ ; \quad \bar{B}': \bar{a}'^{(i)}_k \to \bar{b}_k^{(i)};\ \bar{x}'^{(i)}_k \to \bar{y}_k^{(i)} \ ;$$

выражения для B^- (с заменой букв *a* на *b* и *x* на *y*);

далее (не указываем компоненты набора *B*), $B(t=s) \to (Bt=Bs),\ B^-(Bt=Bs) \to (t=s)$, *t*, *s* - термы.

При построении, чтобы не выйти из *B*-копии формальной склейки, необходимо согласовать набор операций *B* с имеющимися в склейке и её копии функциональными буквами.

Для проекций имеем ($a^{(1)}=C^{(1)}x^{(1)}$, $Ba^{(1)}=BC^{(1)}x^{(1)}$, $Ba^{(1)}=BC^{(1)}B^{-}Bx^{(1)}$, $b'^{(1)}=BC^{(1)}B^{-}y'^{(1)}$, $G'^{(1)}\stackrel{\text{def}}{=}BC^{(1)}B^{-}$ и т.д.)

$$G'^{(1)}\stackrel{\text{def}}{=}BC^{(1)}B^{-},\ G'^{(2)}\stackrel{\text{def}}{=}BC^{(2)}B^{-},\ G^{(1)}\stackrel{\text{def}}{=}BC'^{(1)}B^{-},\ G^{(2)}\stackrel{\text{def}}{=}BC'^{(2)}B^{-},$$

$$\bar{G}'^{(1)}\stackrel{\text{def}}{=}B\bar{C}^{(1)}B^{-},\ \bar{G}'^{(2)}\stackrel{\text{def}}{=}B\bar{C}^{(2)}B^{-},\ \bar{G}^{(1)}\stackrel{\text{def}}{=}B\bar{C}'^{(1)}B^{-},\ \bar{G}^{(2)}\stackrel{\text{def}}{=}B\bar{C}'^{(2)}B^{-}.$$

Для согласования функциональных символов ξ_{ij} и η_{ij} удобно использовать в обозначениях целые индексы от 1 до 8. Рассмотрим четыре случая, первый из которых отвечает выбору индексов $i, j\in\{1,2,5,6\}$. Для $x^{(i)}=\xi_{ij}x^{(j)}$ имеем

$$Bx^{(i)}=B\xi_{ij}B^{-}Bx^{(j)};\ y^{(i+2)}=B\xi_{ij}B^{-}y^{(j+2)};\ y^{(i+2)}=\eta_{i+2,j+2}y^{(j+2)};\ \eta_{i+2,j+2}=B\xi_{ij}B^{-}.$$

Аналогично для остальных случаев.

При $i\in\{1,2,5,6\}$, $j\in\{3,4,7,8\}$ получаем $\eta_{i+2,j-2}=B\xi_{ij}B^{-}$.

При $j\in\{1,2,5,6\}$, $i\in\{3,4,7,8\}$ получаем $\eta_{i-2,j+2}=B\xi_{ij}B^{-}$.

При $i, j\in\{3,4,7,8\}$ получаем $\eta_{i-2,j-2}=B\xi_{ij}B^{-}$.

Дополняем, как и в предыдущем случае, конструктивные операции *B* и *R* до операций $\hat{B}$ и $\hat{R}$, включающих преобразования $\hat{B}:t_1=t_2\to Bt_1=Bt_2$, $\hat{R}:t_1=t_2\to Rt_1=Rt_2$ для термов s_0 сортов и $\hat{B}:A\wedge H\to\hat{B}A\wedge\hat{B}H$, $\hat{R}:A\wedge H\to\hat{R}A\wedge\hat{R}H$ для формул с конъюнкциями.

Операция перевода исходных формул сортов s_0 в формулы сортов s_1 *B*-копии склейки состоит в ***одновременном*** применении всего набора операций из $\hat{B}$.

Отметим, что в исходной склейке свёртками являются не отдельные связные гребни соответствующих сортов, а их тройки, в то время как определены с помощью операций ξ_{ij} отдельные связные гребни свёрточных сортов. Удобно ***свёртками*** считать любые элементы *B*-копии склейки, а в ***свёрточных операциях*** считать используемыми лишь фиксированные наборы свёрток.

Переход от гребней к свёрткам (и обратно) в сам***о***й склейке - обозначим соответствуюшую конструктивную операцию через ξ (ξ^{-} - обозначение для обратной операции) - определён с помощью операций $\xi_{ij}:\xi_{31},\xi_{42},\xi_{75},\xi_{86}(\xi_{13},\xi_{24},\xi_{57},\xi_{68})$, именно,

$$\xi x \to x',\ \xi^{-} x' \to x;\ \xi: a_1^{(1)} = C^{(1)} x_1^{(1)} \wedge a_1^{(2)} = C^{(2)} x_1^{(2)} \wedge x_1^{(1)} = x_1^{(2)} \to$$
$$\to a'^{(1)}_1 = C'^{(1)} x'^{(1)}_1 \wedge a'^{(2)}_1 = C'^{(2)} x'^{(2)}_1 \wedge x'^{(1)}_1 = x'^{(2)}_1,$$
$$a'^{(1)}_1 = \xi_{31} a_1^{(1)},\ x'^{(1)}_1 = \xi_{31} x_1^{(1)},\ a'^{(2)}_2 = \xi_{42} a_2^{(2)},\ x'^{(2)}_1 = \xi_{42} x_1^{(2)},\ C'^{(1)} = \xi_{31} C^{(1)} \xi_{13},$$
$$a_1^{(1)} = \xi_{13} a'^{(1)}_1,\ x_1^{(1)} = \xi_{13} x'^{(1)}_1,\ a_2^{(2)} = \xi_{24} a'^{(2)}_2,\ x_1^{(2)} = \xi_{24} x'^{(2)}_1,\ C^{(1)} = \xi_{13} C'^{(1)} \xi_{31}, \ldots$$

Одновременное применение к внешним гребню и свёртке операций ξ и ξ^{-} обозначим через $\tilde{\xi}$, будучи дважды примененной к формуле операция ничего в ней не меняет ($\tilde{\xi}\tilde{\xi}F$ графически равно F).

Аналогичным образом вводим операцию ζ, расставляющую (и снимающую) надчеркивание, которую обозначим (ζ^{-} - обозначение для обратной операции «стирания» надчеркивания), одновременное применение к формуле обеих операций ζ, ζ^{-} обозначим через $\tilde{\zeta}$. Применение дважды к формуле F и этой операции оставляет формулу неизменной: $\tilde{\zeta}\tilde{\zeta}F$ графически равно F. Операции ξ и ζ «коммутируют», результаты от применения $\xi\zeta$ и $\zeta\xi$ совпадают.

Отметим, что при определении произведений введённые операции фактически используются (в операциях M_b и M с другими индексами), воздействуя на формулы с равенствами проекций.

Зададим тернарное произведение ρ гребней x_1, x_2, x_3

$$a_1^{(1)} = C^{(1)} x_1^{(1)} \wedge a_1^{(2)} = C^{(2)} x_1^{(2)} \wedge x_1^{(1)} = x_1^{(2)}$$
$$a_2^{(1)} = C^{(1)} x_2^{(1)} \wedge a_2^{(2)} = C^{(2)} x_2^{(2)} \wedge x_2^{(1)} = x_2^{(2)}$$
$$a_3^{(1)} = C^{(1)} x_3^{(1)} \wedge a_3^{(2)} = C^{(2)} x_3^{(2)} \wedge x_3^{(1)} = x_3^{(2)}$$

с помощью свёртки из двух гребней свёрток u', v'

$$a'^{(1)}_u = C'^{(1)} u'^{(1)} \wedge a'^{(2)}_u = C'^{(2)} u'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge a'^{(1)}_u = a_1^{(2)} \wedge a'^{(2)}_u = a_2^{(1)} \wedge$$
$$\wedge a'^{(1)}_v = C'^{(1)} v'^{(1)} \wedge a'^{(2)}_v = C'^{(2)} v'^{(2)} \wedge v'^{(1)} = v'^{(2)} a'^{(1)}_v = a_2^{(2)} \wedge a'^{(2)}_v = a_3^{(1)}.$$

В качестве внешнего гребня получим гребень x

$$a_1^{(1)} = C^{(1)} \rho^{(1)} x_1^{(1)} u'^{(1)} x_2^{(1)} v'^{(1)} x_3^{(1)} \wedge a_3^{(2)} = C^{(2)} \rho^{(2)} x_1^{(2)} u'^{(2)} x_2^{(2)} v'^{(2)} x_3^{(2)},$$
$$\rho^{(1)} x_1^{(1)} u'^{(1)} x_2^{(1)} v'^{(1)} x_3^{(1)} = \rho^{(2)} x_1^{(2)} u'^{(2)} x_2^{(2)} v'^{(2)} x_3^{(2)}$$

Итак, $x = \rho x_1 u' x_2 v' x_3$.

Теперь с помощью ρ определим тернарное произведение ρ_T отдельных связных свёрток

$$\xi x = x',\ x' = \xi \rho x_1 u' x_2 v' x_3,\ x' = \xi \rho \xi^{-} x'_1 \xi u \xi^{-} x'_2 \xi v \xi^{-} x'_3,$$
$$x' = \xi \rho \xi^{-}_I x'_1 \xi_{II} u \xi^{-}_{III} x_2 \xi_{IV} v,\ \xi^{-}_V x_3,\ x' = \xi \rho \xi^{-}_I \xi_{II} \xi^{-}_{III} \xi_{IV} \xi^{-}_V x'_1 u x'_2 v x'_3,$$
$$\rho_T = \xi \rho \xi^{-}_I \xi_{II} \xi^{-}_{III} \xi_{IV} \xi^{-}_V.$$

Имеем $x' = \rho_T x'_1 u x'_2 v x'_3 = \xi \rho x_1 u' x_2 v' x_3.$

Итак, для вычисления тернарного произведения свёрток достаточно перевести их в гребни, для трёх гребней вычислить произведение и перейти от результата к соответствующей свёртке.

Отметим, что $x=\xi^{-}x'=\xi^{-}\rho_T x'_1 u x'_2 v x'_3$ есть бинарное произведение гребней *u*, *v*, которым мы пользовались, а $x'=\xi\rho x_1 u' x_2 v' x_3$. есть соответствующее бинарное произведение свёрток *u*', *v*'.

Рассмотрим детальнее связь тернарного и бинарного произведений гребней. Для этого проведём вычисление тернарной операции для трёх гребней *u, w, v*

$$a_u^{(1)}=C^{(1)}u^{(1)}\wedge a_u^{(2)}=C^{(2)}u^{(2)}\wedge u^{(1)}=u^{(2)},$$
$$a_w^{(1)}=C^{(1)}w^{(1)}\wedge a_w^{(2)}=C^{(2)}w^{(2)}\wedge w^{(1)}=w^{(2)},$$
$$a_v^{(1)}=C^{(1)}v^{(1)}\wedge a_v^{(2)}=C^{(2)}v^{(2)}\wedge v^{(1)}=v^{(2)}$$

и двух свёрток

$$a'^{(1)}_1=C'^{(1)}x'^{(1)}_1\wedge a'^{(2)}_1=C'^{(2)}x'^{(2)}_1\wedge x'^{(1)}_1=x'^{(2)}_1\wedge a'^{(1)}_1=a_u^{(2)}\wedge a'^{(2)}_1=a_w^{(1)}\wedge$$
$$\wedge a'^{(1)}_2=C'^{(1)}x'^{(1)}_2\wedge a'^{(2)}_2=C'^{(2)}x'^{(2)}_2\wedge x'^{(1)}_2=x'^{(2)}_2\wedge a'^{(1)}_2=a_w^{(2)}\wedge a'^{(2)}_2=a_v^{(1)}.$$

После проведения операции M_b и операции M_l получаем

$a_u^{(1)}=C^{(1)}u^{(1)}$ $a_v^{(2)}=C^{(2)}v^{(2)}$	$\varnothing$
$\bar{a}_1^{(1)}=\bar{C}^{(1)}\bar{x}_1^{(1)}\wedge\bar{a}_1^{(2)}=\bar{C}^{(2)}\bar{x}_1^{(2)}$ $\wedge\bar{a}_2^{(1)}=\bar{C}^{(1)}\bar{x}_2^{(1)}\wedge\bar{a}_2^{(2)}=\bar{C}^{(2)}\bar{x}_2^{(2)}$ $\bar{a}_u^{(2)}=\bar{C}^{(2)}\bar{u}^{(2)}$ $\bar{a}_v^{(1)}=\bar{C}^{(1)}\bar{v}'^{(1)}$ $\bar{a}_w^{(1)}=\bar{C}^{(1)}\bar{w}^{(1)}\wedge\bar{a}_w^{(2)}=\bar{C}^{(2)}\bar{w}^{(2)}$	$\varnothing$

Это в точности тот же результат, что и при вычислении бинарного призведения с операцией μ выше. Применение операции *Ev* теперь даёт

$$\rho u x'_1 w x'_2 v=\lambda u' x_1 w' x_2 v' \quad .$$

Применим ξ к этой формуле получим

$$\xi x=x'=\xi\rho u x'_1 w x'_2 v=\xi\lambda u' x_1 w' x_2 v' \quad ,$$

что даёт, соответственно, бинарную и тернарную операции умножения для свёрток, итак, бинарная операция для гребней переходит в тернарную операцию для свёрток, а тернарная операция для гребней переходит в бинарную операцию для свёрток.

Вернёмся к дуальной по свёрткам склейке.

Операции из $\hat{B}$ переводят свёртку во внешний гребень дуальной склейки, а гребень в свёртку. Конструктивные операции, составляющие процедуру выполнения свёртки для бинарной λ и тернарной операций ρ остаются теми же. Как и в предыдущем случае сортовой дуальности, определим действие бинарной и тернарной операций в *B*-копии, которые обозначим соответственно ν и σ через операции λ и ρ.

Посмотрим, куда перейдет бинарная операция μ, результат *x* ее выполнения переходит в *Bx*. Применяем операцию $\hat{B}$ к гребню

$$a'^{(1)}_u = C'^{(1)} u'^{(1)} \wedge a'^{(2)}_u = C'^{(2)} u'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge a'^{(2)}_u = a^{(1)}_1 \wedge$$
$$\wedge a^{(1)}_1 = C^{(1)} x^{(1)}_1 \wedge a^{(2)}_1 = C^{(2)} x^{(2)}_1 \wedge x^{(1)}_1 = x^{(2)}_1 \wedge a^{(2)}_1 = a'^{(1)}_w \wedge$$
$$\wedge a'^{(1)}_w = C'^{(1)} w'^{(1)} \wedge a'^{(2)}_w = C'^{(2)} w'^{(2)} \wedge w'^{(1)} = w'^{(2)} \wedge a'^{(2)}_w = a^{(1)}_2 \wedge$$
$$\wedge a^{(1)}_2 = C^{(1)} x^{(1)}_2 \wedge a^{(2)}_2 = C^{(2)} x^{(2)}_2 \wedge x^{(1)}_2 = x^{(2)}_2 \wedge a^{(2)}_2 = a'^{(1)}_v \wedge$$
$$\wedge a'^{(1)}_v = C'^{(1)} v'^{(1)} \wedge a'^{(2)}_v = C'^{(2)} v'^{(2)} \wedge v'^{(1)} = v'^{(2)}.$$

Применяем операцию $\hat{B}$, имеем

$$b^{(1)}_p = G^{(1)} p^{(1)} \wedge b^{(2)}_p = G^{(2)} p^{(2)} \wedge p^{(1)} = p^{(2)} \wedge b^{(2)}_p = b'^{(1)}_1 \wedge$$
$$\wedge b'^{(1)}_1 = G'^{(1)} y'^{(1)}_1 \wedge b'^{(2)}_1 = G'^{(2)} y'^{(2)}_1 \wedge y'^{(1)}_1 = y'^{(2)}_1 \wedge b'^{(2)}_1 = b^{(1)}_q \wedge$$
$$\wedge b^{(1)}_q = G^{(1)} q^{(1)} \wedge b^{(2)}_q = G^{(2)} q^{(2)} \wedge q^{(1)} = q^{(2)} \wedge b^{(2)}_q = b'^{(1)}_2 \wedge$$
$$\wedge b'^{(1)}_2 = G'^{(1)} y'^{(1)}_2 \wedge b'^{(2)}_2 = G'^{(2)} y'^{(2)}_2 \wedge y'^{(1)}_2 = y'^{(2)}_2 \wedge b'^{(2)}_2 = b^{(1)}_r \wedge$$
$$\wedge b^{(1)}_r = G^{(1)} r^{(1)} \wedge b^{(2)}_r = G^{(2)} r^{(2)} \wedge r^{(1)} = r^{(2)}.$$

Поскольку операции в склейке и *B*-копии те же самые, то выполняя их для последнего гребня *B*-копии склейки приходим к результату

$$\hat{B}x = \hat{B}\lambda u' x_1 w' x_2 v',\ y' = \hat{B}\lambda u' x_1 w' x_2 v',\ \hat{B}\lambda u' x_1 w' x_2 v' = \sigma\, p\, y'_1 q\, y'_2 r$$

при $\hat{B}x = y',\ \hat{B}u' = p,\ \hat{B}x_1 = y'_1,\ \hat{B}w' = q,\ \hat{B}x_2 = y'_2\ \hat{B}v' = r.$

Итак, $\hat{B}\lambda u' x_1 w' x_2 v' = \sigma\, p\, y'_1 q\, y'_2 r$, откуда

$$\sigma\, p\, y'_1 q\, y'_2 r = \hat{B}\lambda u' x_1 w' x_2 v' = \hat{B}\lambda\, \hat{B}^{-} p\, \hat{B}^{-} y'_1\, \hat{B}^{-} q\, \hat{B}^{-} y'_2\, \hat{B}^{-} r$$

$$\sigma\, p\, y'_1 q\, y'_2 r \stackrel{\text{def}}{=} \hat{B}\lambda\, \hat{B}^{-}_{I}\, \hat{B}^{-}_{II}\, \hat{B}^{-}_{III}\, \hat{B}^{-}_{IV}\, \hat{B}^{-}_{V}\, p\, y'_1 q\, y'_2 r \quad .$$

Таким образом, вычисление тернарного произведения гребней *p, q, r* сведено к вычислению бинарного произведения x_1 и x_2.

Аналогичным образом получаем

$$\nu\, p'\, y_1 q'\, y_2 r' \stackrel{\text{def}}{=} \hat{B}\rho\, \hat{B}^{-}_{I}\, \hat{B}^{-}_{II}\, \hat{B}^{-}_{III}\, \hat{B}^{-}_{IV}\, \hat{B}^{-}_{V}\, p'\, y_1 q'\, y_2 r' \quad ,$$

то есть вычисление бинарного произведения в *B*-копии склейки сведено к вычислению тернарного произведения исходной склейки.

Определение

Формулы, полученные с помощью набора операций $\hat{B}$, назовём ***дуальными по свёрткам*** к исходным формулам. Возникающую при действии $\hat{B}$ категорную склейку назовём ***дуальной по свёрткам*** к исходной направленной бинарной склейке.

Таким образом, конструктивная часть построения дуальной по свёрткам склейки получена.

Кроме введённых вариантов двойственности D и B можно построить новые двойственности, применив операции D и B несколько раз. Некоторые свойства этих двойственностей рассмотрены далее при введении логики в язык.

8. Категорный принцип двойственности

До данного раздела изложение велось в рамках общесистемной конструктивности [1], обсуждались лишь различные конструктивные операции, с помощью которых строились конструктивные объекты. Тем не менее, введённые операции понадобятся при доказательстве теорем. В используемом языке не вводились высказывания и, соответственно, понятие истинности высказываний. Здесь мы погрузим формальные построения в первопорядковый язык интуиционистской и классической логик, оставаясь в рамках теории доказательств и сделав некоторые замечания о других логиках, в которых будет выполняться обсуждаемый категорный принцип двойственности, аналогичный обычному принципу двойственности в теории категорий.

Для определённости будем опираться в том числе и в многосортном случае на терминологию из [9] и формулировки (аксиомы, правила вывода и так далее) исчисления предикатов CPC и HPC (PC обозначает случай, когда имеются в виду и CPC, и HPC), соответственно, в классической и интуиционистской логике, выписанные в книге [9] (см. стр. 19-20 и другие). Мы используем CPC и понятия теории множеств, но без обсуждения возможных интерпретаций формального языка в множествах. Далее, мы рассмотрим некотрые вопросы теории категорных склеек для случая интуиционистской и некоторых других логик.

Построенный язык можно рассматривать как часть логико-математического языка Ω (о языке Ω см. для определенности [9]) исчисления предикатов с равенством, связками, $\neg$, $\wedge$, $\vee$, $\supset$, кванторами $\forall$, $\exists$ и некоторыми нелогическими аксиомами. Для этого переменные, функциональные символы и равенство, а также знак конъюнкции отождествляем с соответствующими элементами языка Ω.

Предложение

Введённые выше формулы полных гребней и их проекций, использующие символы равенства и конъюнкции, являются правильно построенными формулами языка Ω.

Предложение

$$\forall a_k^{(i)} \forall x_m^{(j)} \left(a_k^{(i)} = x_m^{(j)} \supset a_k^{(i)} = a_m^{(j)} \right)$$

Приведём для примера простое доказательство (дерево вывода, сорта не указаны) в РС этого утверждения, используя подходящие аксиомы и допустимые правила вывода

$$\dfrac{\dfrac{\dfrac{\dfrac{a_k = x_m \qquad a_k = C\,x_k}{C\,x_k = x_m}}{C\,C\,x_k = C\,x_m \qquad C\,C\,t = C\,t\,(t - терм)}}{C\,x_k = C\,x_m \qquad a_l = C\,x_l}}{a_k = a_m}$$

$$a_k = x_m \vdash a_k = a_m$$

$$\vdash a_k = x_m \supset a_k = a_m$$

Это предложение позволяет в полных гребнях оставить только равенства между проекциями и равенства между переменными (термами).

Не указывая сорта, выпишем условие ассоциативности для операции μ, для бинарной операции λ в формуле следует указать в соответствующих местах штрихи (см. рис. 2)

$$\mu\, u_{12,3} \mu\, u_{12} x_1 w_2 x_2 v_{12} w_{12,3} x_3 v_{12,3} = \mu\, u_{1,23} x_1 w_{1,23} \mu\, u_{23} x_2 w_{23} x_3 v_{23} v_{1,23}$$

Рассмотрим простой пример склейки, порождаемой заданной категорией предпорядка. Для склейки, порождаемой произвольной категорией, стрелка с именем x_i сопоставляется гребню $a_i^{(1)} = C^{(1)} x_i^{(1)} \wedge a_i^{(2)} = C^{(2)} x_i^{(2)} \wedge x_i^{(1)} = x_i^{(2)}$, начало стрелки сопоставляется $a_i^{(1)}$, конец стрелки сопоставляется $a_i^{(2)}$. В свёртки помимо ξ_{21}-образов категорных единичных стрелок объектов могут входить ξ_{21}-образы других стрелок. Операция λ выражается через композицию в категории $x_3^{(j)} = \lambda^{(j)} u'^{(j)} x_1^{(j)} w'^{(j)} x_2^{(j)} v'^{(j)} = u^{(j)} \circ x_1^{(j)} \circ w^{(j)} \circ x_2^{(j)} \circ v^{(j)}$, $u = \xi_{12} u'$, $w = \xi_{12} w'$, $v = \xi_{12} v'$ - стрелки категории, входящие в определение свёртки для гребней.

Предложение

Операция λ является ассоциативной по аргументам x_1 и x_2 лишь при дополнительных условиях. Для склейки, порождаемой заданной категорией предпорядка, указанные условия выписаны в доказательстве предложения.

Доказательство

Рассмотрим схему рис. 2. Каждое её ребро изображает гребень или свёртку (пунктир) произведения, μ_{12} - внешний гребень произведения x_1 на x_2, μ_{23} - внешний гребень произведения x_2 на x_3 .

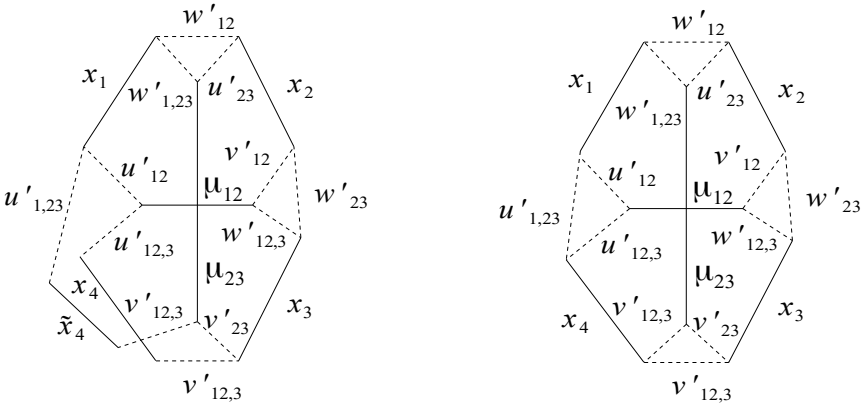


Рис. 2. Схемы для двух возможных произведений трёх гребней.

Схема предполагает (сорта не указаны), что x_4 получается двумя способами

$$x_4=\mu\, u_{12,3}\mu\, u_{12}\, x_1\, w_{12}\, x_2\, v_{12} w_{12,3}\, x_3\, v_{12,3}=u_{12,3}\circ u_{12}\circ x_1\circ w_{12}\circ x_2\circ v_{12}\circ w_{12,3}\circ x_3\circ v_{12,3};$$
$$\tilde{x}_4=\mu\, u_{1,23}\, x_1\, w_{1,23}\mu\, u_{23} x_2 w_{23}\, x_3 v_{23} v_{1,23}=u_{1,23}\circ x_1\circ w_{1,23}\circ u_{23}\circ x_2\circ w_{23}\circ x_3\circ v_{23}\circ v_{1,23}.$$

Условие ассоциативности операции μ по аргументам x_1 и x_2 имеет, как видно из приведённых формул, вид

$$x_4=\mu\, u_{12,3}\mu\, u_{12}\, x_1\, w_2\, x_2\, v_{12}\, w_{12,3}\, x_3\, v_{12,3}=\tilde{x}_4=\mu\, u_{1,23}\, x_1\, w_{1,23}\mu\, u_{23}\, x_2\, w_{23} x_3\, v_{23}\, v_{1,23}. \quad (10)$$

В категориях предпорядка стрелок между объектами не более одной, поэтому можно заключить, что

$$w_{12}=w_{1,23}\circ u_{23};\ w_{23}=v_{12}\circ w_{12,3}.$$

Последних равенств для ассоциативности произведения в склейке надостаточно.

Если λ ассоциативна и μ ассоциативна по аргументам x_1 и x_2, то из предыдущих равенств и (10) следует, что (через F обозначен одинаковый фрагмент в формулах)

$$x_4=u_{12,3}\circ u_{12}\circ F(x_1, x_2, x_3)\circ v_{12,3}=\tilde{x}_4=u_{1,23}\circ F(x_1, x_2, x_3)\circ v_{23}\circ v_{1,23}.$$

В категориях предпорядка эти равенства выполняются, так как ввиду единственности стрелки между объектами имеем

$$u_{1,23}=u_{12,3}\circ u_{12};\ v_{12,3}=v_{23}\circ v_{1,23},$$

что согласуется с ассоциативностью произведения (см. правую схему на рис. 2).

Предложение доказано.

Последние равенства для произвольных категорий отражают наличие ограничений на операцию μ.

Гребни склейки вместе с внешним и внутренним гребнями конструктивно получаются применением операций умножения к цепочкам нерва склейки, в которые входят лишь внешние гребни и свёртки.

Определение

Категорной направленной бинарной склейкой называется множество всех полных гребней, которые можно получить из базовых гребней с помощью операций свёртки при наличии нелогических аксиом

$$(1a)\quad \begin{gathered} C_1 x_m^{(1)} \stackrel{\text{def}}{=} \xi_{1,2k+1} C^{(2k+1)} \xi_{2k+1,1} x_m^{(1)},\ C_2 x_m^{(1)} \stackrel{\text{def}}{=} \xi_{1,2q} C^{(2k)} \xi_{2q,1} x_m^{(1)}, \\ \forall x_m^{(1)} C_i C_j x_m^{(1)} = C_j x_m^{(1)}, \\ \forall x_m^{(\alpha)} \left(\xi_{\alpha\beta} \xi_{\beta\alpha} x_m^{(\alpha)} = x_m^{(\alpha)} \right), \\ i, j = 1,2, k = 0,1,2,3, q = 1,2,3,4, \alpha, \beta = 1, \ldots, 8, m = 1,2,3, \ldots \end{gathered}$$

(далее нижние индексы у переменных *x, y,..., w, v* опущены)

$$(2a)\quad \begin{gathered} \forall x^{(1)} \forall y^{(1)} \forall w'^{(1)} \forall u'^{(1)} \forall v'^{(1)} \forall x^{(2)} \forall y^{(2)} \forall w'^{(2)} \forall u'^{(2)} \forall v'^{(2)} \\ x^{(1)} = x^{(2)} \wedge y^{(1)} = y^{(2)} \wedge w'^{(1)} = w'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge v'^{(1)} = v'^{(2)} \wedge \\ \wedge \exists z^{(1)} \exists z^{(2)} \left(\lambda^{(1)} u'^{(1)} x^{(1)} w'^{(1)} y^{(1)} v'^{(1)} = z^{(1)} \wedge \lambda^{(2)} u'^{(2)} x^{(2)} w'^{(2)} y^{(2)} v'^{(2)} = z^{(2)} \right) \equiv \\ \equiv C'^{(2)} u'^{(2)} = C^{(1)} x^{(1)} \wedge C'^{(1)} w'^{(1)} = C^{(2)} x^{(2)} \wedge \\ \wedge C'^{(2)} w'^{(2)} = C^{(1)} y^{(1)} \wedge C'^{(1)} v'^{(1)} = C^{(2)} y^{(2)}, \end{gathered}$$

$$(2a')\quad \begin{gathered} \forall x'^{(1)} \forall y'^{(1)} \forall w^{(1)} \forall u^{(1)} \forall v^{(1)} \forall x'^{(2)} \forall x'^{(2)} \forall y^{(2)} \forall w^{(2)} \forall u^{(2)} \forall v^{(2)} \\ x'^{(1)} = x'^{(2)} \wedge y'^{(1)} = y'^{(2)} \wedge w^{(1)} = w^{(2)} \wedge u^{(1)} = u^{(2)} \wedge v^{(1)} = v^{(2)} \wedge \\ \exists z^{(1)} \exists z^{(2)} \left(\rho^{(1)} u^{(1)} x'^{(1)} w^{(1)} y'^{(1)} v^{(1)} = z^{(1)} \wedge \rho^{(2)} u^{(2)} x'^{(2)} w^{(2)} y'^{(2)} v^{(2)} = z^{(2)} \right) \equiv \\ \equiv C^{(2)} u^{(2)} = C'^{(1)} x'^{(1)} \wedge C^{(1)} w^{(1)} = C'^{(2)} x'^{(2)} \wedge \\ \wedge C^{(2)} w^{(2)} = C'^{(1)} y'^{(1)} \wedge C^{(1)} v^{(1)} = C'^{(2)} y'^{(2)}, \end{gathered}$$

$$(3a)\quad \begin{gathered} \forall x^{(1)} \forall y^{(1)} \forall w'^{(1)} \forall u'^{(1)} \forall v'^{(1)} \forall x^{(2)} \forall y^{(2)} \forall w'^{(2)} \forall u'^{(2)} \forall v'^{(2)} \forall z^{(1)} \forall z^{(2)} \\ x^{(1)} = x^{(2)} \wedge y^{(1)} = y^{(2)} \wedge w'^{(1)} = w'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge v'^{(1)} = v'^{(2)} \wedge \\ \wedge \left(\lambda^{(1)} u'^{(1)} x^{(1)} w'^{(1)} y^{(1)} v'^{(1)} = z^{(1)} \wedge \lambda^{(2)} u'^{(2)} x^{(2)} w'^{(2)} y^{(2)} v'^{(2)} = z^{(2)} \right) \supset \\ \supset \left(C'^{(1)} u'^{(1)} = C^{(1)} z^{(1)} \wedge C'^{(2)} v'^{(2)} = C^{(2)} z^{(2)} \right) \end{gathered}$$

$$(3a')\quad \begin{gathered} \forall x'^{(1)} \forall y'^{(1)} \forall w^{(1)} \forall u^{(1)} \forall v^{(1)} \forall x'^{(2)} \forall y'^{(2)} \forall w^{(2)} \forall u^{(2)} \forall v^{(2)} \forall z^{(1)} \forall z^{(2)} \\ x'^{(1)} = x'^{(2)} \wedge y'^{(1)} = y'^{(2)} \wedge w^{(1)} = w^{(2)} \wedge u^{(1)} = u^{(2)} \wedge v^{(1)} = v^{(2)} \wedge z^{(1)} = z^{(2)} \wedge \\ \wedge \left(\rho^{(1)} u^{(1)} x'^{(1)} w^{(1)} y'^{(1)} v^{(1)} = z^{(1)} \wedge \rho^{(2)} u^{(2)} x'^{(2)} w^{(2)} y'^{(2)} v^{(2)} = z^{(2)} \right) \supset \\ \supset \left(C^{(1)} u^{(1)} = C^{(1)} z^{(1)} \wedge C^{(2)} v^{(2)} = C^{(2)} z^{(2)} \right) \end{gathered}$$

$$
\begin{gathered}
\forall x^{(1)}\forall y^{(1)}\forall w^{(1)}\forall u^{(1)}\forall v^{(1)}\forall x^{(2)}\forall y^{(2)}\forall w^{(2)}\forall u^{(2)}\forall v^{(2)}\\
\forall x'^{(1)}\forall y'^{(1)}\forall w'^{(1)}\forall u'^{(1)}\forall v'^{(1)}\forall x'^{(2)}\forall y'^{(2)}\forall w'^{(2)}\forall u'^{(2)}\forall v'^{(2)}\\
x^{(1)}=x^{(2)}\wedge y^{(1)}=y^{(2)}\wedge w'^{(1)}=w'^{(2)}\wedge u'^{(1)}=u'^{(2)}\wedge v'^{(1)}=v'^{(2)}\wedge\\
\wedge x'^{(1)}=x'^{(2)}\wedge y'^{(1)}=y'^{(2)}\wedge w^{(1)}=w^{(2)}\wedge u^{(1)}=u^{(2)}\wedge v^{(1)}=v^{(2)}\wedge\\
\wedge\lambda u'^{(1)}x^{(1)}w'^{(1)}y^{(1)}v'^{(1)}=\rho^{(1)}u^{(1)}x'^{(1)}w^{(1)}y'^{(1)}v^{(1)}\wedge\\
\wedge\lambda u'^{(2)}x^{(2)}w'^{(2)}y^{(2)}v'^{(2)}=\rho^{(2)}u^{(2)}x'^{(2)}w^{(2)}y'^{(2)}v^{(2)}\wedge\\
\wedge x'^{(1)}=\xi_{31}x^{(1)}\wedge y'^{(1)}=\xi_{31}y^{(1)}\wedge w'^{(1)}=\xi_{31}w^{(1)}\wedge u'^{(1)}=\xi_{31}u^{(1)}\wedge v'^{(1)}=\xi_{31}v^{(1)}\wedge\\
\wedge x'^{(2)}=\xi_{31}x^{(2)}\wedge y'^{(2)}=\xi_{31}y^{(2)}\wedge w'^{(2)}=\xi_{31}w^{(2)}\wedge u'^{(2)}=\xi_{31}u^{(2)}\wedge v'^{(2)}=\xi_{31}v^{(2)}.
\end{gathered}
\tag{4a}
$$

Ассоциативность для склеек имеет смысл для операций λ с разными параметрами. Для ***ассоциативной*** категорной направленной бинарной склейки имеет место дополнительная аксиома

$$
\begin{gathered}
\forall x_1^{(1)}\forall x_2^{(1)}\forall x_3^{(1)}\forall x_4^{(1)}\forall x_5^{(1)}\forall x_6^{(1)}\forall x_7^{(1)}\forall x_1^{(2)}\forall x_2^{(2)}\forall x_3^{(2)}\forall x_4^{(2)}\forall x_5^{(2)}\forall x_6^{(2)}\forall x_7^{(2)}\\
\forall u'^{(1)}_4\forall u'^{(1)}_5\forall u'^{(1)}_6\forall u'^{(1)}_7\forall u'^{(2)}_4\forall u'^{(2)}_5\forall u'^{(2)}_6\forall u'^{(2)}_7\\
\forall w'^{(1)}_4\forall w'^{(1)}_5\forall w'^{(1)}_6\forall w'^{(1)}_7\forall w'^{(2)}_4\forall w'^{(2)}_5\forall w'^{(2)}_6\forall w'^{(2)}_7\\
\forall v'^{(1)}_4\forall v'^{(1)}_5\forall v'^{(1)}_6\forall v'^{(1)}_7\forall v'^{(2)}_4\forall v'^{(2)}_5\forall v'^{(2)}_6\forall v'^{(2)}_7\\
x_1^{(1)}=x_1^{(2)}\wedge x_2^{(1)}=x_2^{(2)}\wedge x_3^{(1)}=x_3^{(2)}\wedge x_4^{(1)}=x_4^{(2)}\wedge x_5^{(1)}=x_5^{(2)}\wedge x_6^{(1)}=x_6^{(2)}\wedge x_7^{(1)}=x_7^{(2)}\wedge\\
\wedge u'^{(1)}_4=u'^{(2)}_4\wedge v'^{(1)}_4=v'^{(2)}_4\wedge w'^{(1)}_4=w'^{(2)}_4\wedge u'^{(1)}_5=u'^{(2)}_5\wedge v'^{(1)}_5=v'^{(2)}_5\wedge w'^{(1)}_5=w'^{(2)}_5\wedge\\
\wedge u'^{(1)}_6=u'^{(2)}_6\wedge v'^{(1)}_6=v'^{(2)}_6\wedge w'^{(1)}_6=w'^{(2)}_6\wedge u'^{(1)}_7=u'^{(2)}_7\wedge v'^{(1)}_7=v'^{(2)}_7\wedge w'^{(1)}_7=w'^{(2)}_7\wedge\\
\wedge x_4^{(1)}=\lambda^{(1)}u'^{(1)}_4x_1^{(1)}w'^{(1)}_4x_2^{(1)}v'^{(1)}_4\wedge x_4^{(2)}=\lambda^{(2)}u'^{(2)}_4x_1^{(2)}w'^{(2)}_4x_2^{(2)}v'^{(2)}_4\wedge\\
\wedge x_5^{(1)}=\lambda^{(1)}u'^{(1)}_5x_2^{(1)}w'^{(1)}_5x_3^{(1)}v'^{(1)}_5\wedge x_5^{(2)}=\lambda^{(2)}u'^{(2)}_5x_2^{(2)}w'^{(2)}_5x_3^{(2)}v'^{(2)}_5\wedge\\
\wedge x_6^{(1)}=\lambda^{(1)}u'^{(1)}_6x_1^{(1)}w'^{(1)}_6x_5^{(1)}v'^{(1)}_6\wedge x_6^{(2)}=\lambda^{(2)}u'^{(2)}_6x_1^{(2)}w'^{(2)}_6x_5^{(2)}v'^{(2)}_6\wedge\\
\wedge x_7^{(1)}=\lambda^{(1)}u'^{(1)}_7x_1^{(1)}w'^{(1)}_7x_2^{(1)}v'^{(1)}_7\wedge x_7^{(2)}=\lambda^{(2)}u'^{(2)}_7x_1^{(2)}w'^{(2)}_7x_2^{(2)}v'^{(2)}_7\supset\\
\supset x_6^{(1)}=x_7^{(1)}\wedge x_6^{(2)}=x_7^{(2)}.
\end{gathered}
\tag{5a}
$$

Не указывая сортов, (5*a*) можно записать через операцию λ (кванторы всеобщности подразумеваются)

$$\lambda u'_7\lambda u'_4x_1w'_4x_2v'_4w'_7x_3v'_7=\lambda u'_6x_1w'_6\lambda u'_5x_2w'_5x_3v'_5v'_6.$$

Применяя операцию $\hat{R}$ к (1*a*-5*a*) мы получаем необходимые аксиомы для объединенного исходного языка с его *R*-копией.

В [6], где задан язык первого порядка в рамках СРС для теории категорий, под категорией понимается теоретико-множественная модель языка, в то время как формальный язык задаёт понятие абстрактной категории, аналогично в [5] используется термин метакатегории, вместе с терминами стрелка, композиция, единица и так далее. Мы не рассматриваем

моделей, вводимые понятия относятся к формальной теории, как и в [5-6] для случая абстрактных категорий и метакатегорий.

Теорема

Категория является частным случаем направленной бинарной категорной склейки.

Для несложного доказательства (которое мы не приводим из-за громоздскости) достаточно в ассоциативной склейке выбрать подходящие частные случаи свёрток, считать проекции единичными стрелками, а также указать перевод многосортного языка в односортный язык теории категорий (например, из [6]).

В самой категории через обычное умножение стрелок можно, перенося отдельные стрелки в свёртки, получать ряд примеров бинарных направленных склеек, порождаемых категорией.

Принцип двойственности касается двойственных понятий в категориях. Аналогичные понятия имеются в категорных склейках. Функтор двойственности меняет местами входы и выходы, а также места вхождений в операцию свёртки сомножителей.

Определим начальные и конечные объекты в склейках, дающие простой, но важный пример двойственных понятий, которым мы, заодно, проиллюстрируем сортовую двойственность.

Определение

Определим ***объект*** склейки как гребень с дополнительным условием

$$a^{(1)}=C^{(1)}x^{(1)}\wedge a^{(2)}=C^{(2)}x^{(2)}\wedge a^{(1)}=x^{(1)} \quad .$$

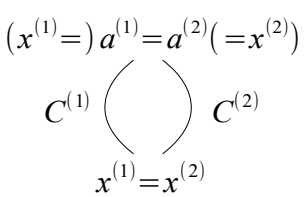
Началом гребня $a^{(1)}=C^{(1)}x^{(1)}\wedge a^{(2)}=C^{(2)}x^{(2)}$ называется объект

$$\alpha^{(1)}=C^{(1)}\lambda^{(1)}\wedge\alpha^{(2)}=C^{(2)}\lambda^{(2)}\wedge\alpha^{(1)}=\lambda^{(1)}\wedge\alpha^{(1)}=a^{(1)}\wedge C^{(1)}x^{(1)}=a^{(1)} \quad ,$$

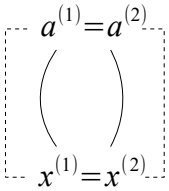
концом этого гребня называется объект

$$\beta^{(1)}=C^{(1)}\sigma^{(1)}\wedge\beta^{(2)}=C^{(2)}\sigma^{(2)}\wedge\beta^{(2)}=\sigma^{(1)}\wedge\beta^{(1)}=a^{(2)}\wedge C^{(2)}x^{(2)}=a^{(2)} \quad .$$

В силу следующего простого предложения графическое выражение для объекта имеет вид



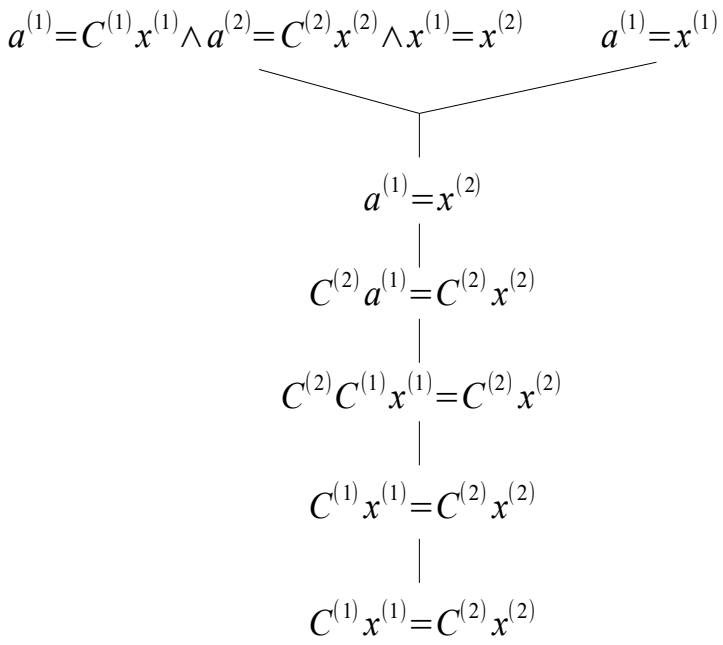
или в сокращении



Предложение

$$a^{(1)}=C^{(1)}x^{(1)}\wedge a^{(2)}=C^{(2)}x^{(2)}\wedge x^{(1)}=x^{(2)},\ x^{(1)}=a^{(1)}\ |\!-\ a^{(1)}=a^{(2)} \quad .$$

Действительно, имеем следующий вывод (переменные переходов между сортами не указаны) для предложения,



что доказывает предложение.

Определение

Начальный объект α для склеек $\alpha^{(1)}=C^{(1)}u^{(1)}\wedge\alpha^{(2)}=C^{(2)}u^{(2)}\wedge\alpha^{(1)}=u^{(1)}$ определяется условием

$$\forall a^{(2)}\exists!\, x^{(1)}\exists!\, x^{(2)}\left(\alpha^{(1)}=C^{(1)}x^{(1)}\wedge a^{(2)}=C^{(2)}x^{(2)}\right) .$$

Конечный объект β для склеек $\beta^{(1)}=C^{(1)}v^{(1)}\wedge\beta^{(2)}=C^{(2)}v^{(2)}\wedge\beta^{(1)}=v^{(1)}$ определяется условием

$$\forall a^{(1)}\exists!\, x^{(1)}\exists!\, x^{(2)}\left(a^{(1)}=C^{(1)}x^{(1)}\wedge\beta^{(2)}=C^{(2)}x^{(2)}\right) .$$

Определение закончено.

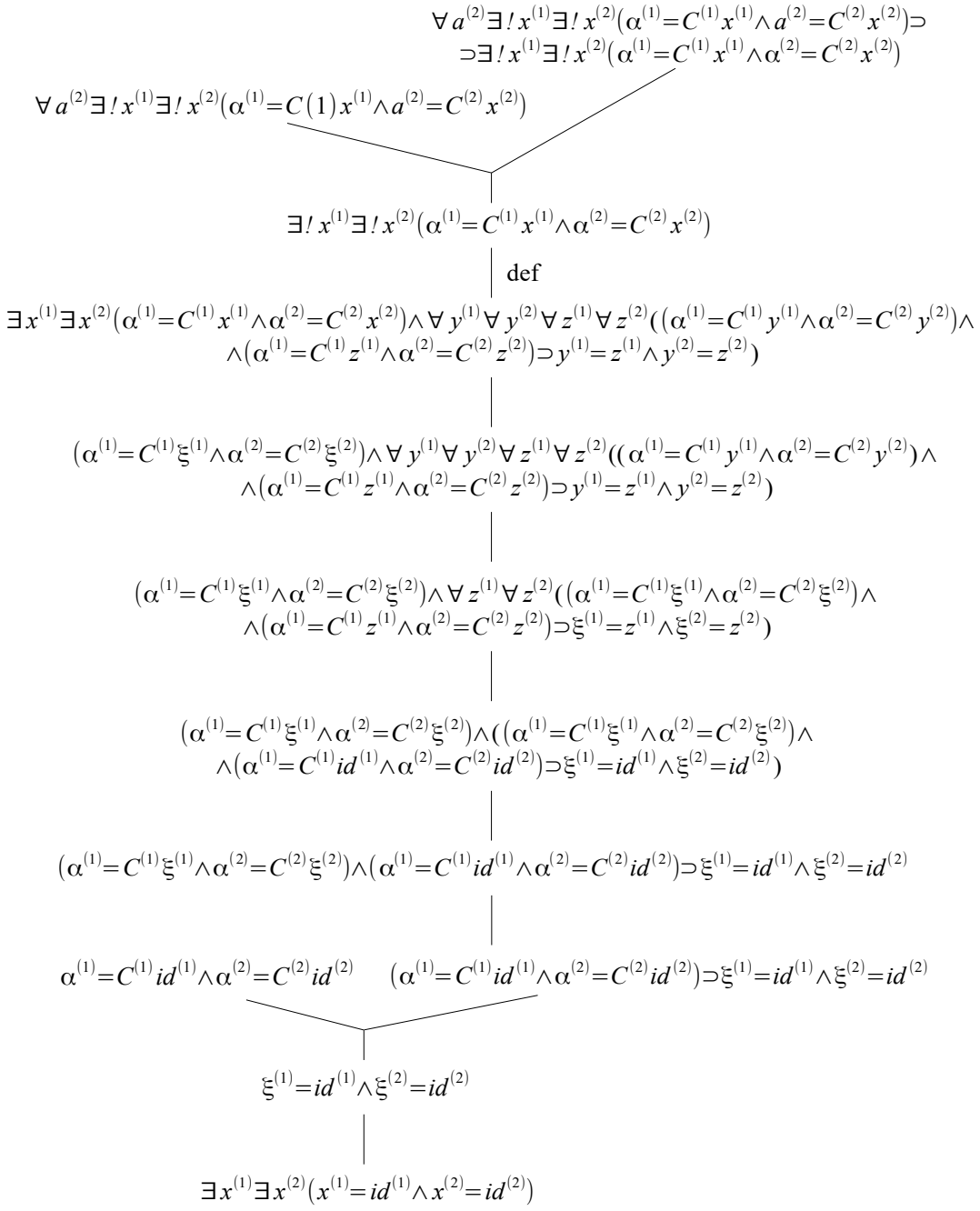
В определении не фигурируют операции умножения гребней, оно имеет смысл независимо от операций.

Как и в теории категорий один объект может иметь несколько одновременно исходящих и входящих в этот объект гребней. Интуитивно очевидно, что начальный и конечный объекты имеют всего один такой гребень. Строго говоря, это утверждение требует формального доказательства, которое будет давать также обоснование согласованности определений этих объектов.

Предложение

$\forall a^{(2)}\exists!\, x^{(1)}\exists!\, x^{(2)}\left(\alpha^{(1)}=C^{(1)}x^{(1)}\wedge a^{(2)}=C^{(2)}x^{(2)}\right)|-$
$|-\exists x^{(1)}\exists x^{(2)}\left(x^{(1)}=id^{(1)}\wedge x^{(2)}=id^{(2)}\right)$

Доказательство даётся следующим выводом (через id обозначен объект α).



Что и требовалось.

Введём понятие изоморфизма объектов склеек.

Определение

Гребень x, имеющий α начало

$$\alpha^{(1)}=C^{(1)}\lambda^{(1)}\wedge\alpha^{(2)}=C^{(2)}\lambda^{(2)}\wedge\alpha^{(1)}=\lambda^{(1)}\wedge\alpha^{(1)}=a^{(1)}\wedge C^{(1)}x^{(1)}=a^{(1)}$$

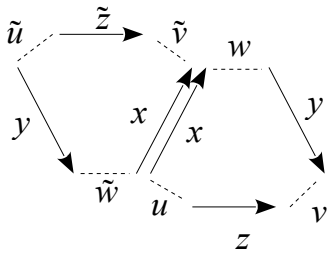
и β конец

$$\beta^{(1)}=C^{(1)}\sigma^{(1)}\wedge\beta^{(2)}=C^{(2)}\sigma^{(2)}\wedge\beta^{(2)}=\sigma^{(1)}\wedge\beta^{(1)}=a^{(2)}\wedge C^{(2)}x^{(2)}=a^{(2)}\quad,$$

называется ***обратимым***, если существует гребень y и $w^{(i)}, u^{(i)}, v^{(i)}, \tilde{w}^{(i)}, \tilde{u}^{(i)}, \tilde{v}^{(i)}, i=1,2$ для операции умножения, такие, что

$$\exists y^{(1)} \exists y^{(2)} \exists z^{(1)} \exists z^{(2)} \exists \tilde{z}^{(1)} \exists \tilde{z}^{(2)}$$
$$(\mu^{(1)} x^{(1)} y^{(1)} w^{(1)} u^{(1)} v^{(1)} = z^{(1)} \wedge \mu^{(2)} x^{(2)} y^{(2)} w^{(2)} u^{(2)} v^{(2)} = z^{(2)} \wedge$$
$$\wedge z^{(1)} = z^{(2)} \wedge C^{(1)} z^{(1)} = \alpha^{(2)} \wedge C^{(2)} z^{(2)} = \alpha^{(1)} \wedge \alpha^{(1)} = \alpha^{(2)} \wedge$$
$$\wedge \mu^{(1)} y^{(1)} x^{(1)} \tilde{w}^{(1)} \tilde{u}^{(1)} \tilde{v}^{(1)} = \tilde{z}^{(1)} \wedge \mu^{(2)} y^{(2)} x^{(2)} \tilde{w}^{(2)} \tilde{u}^{(2)} \tilde{v}^{(2)} = \tilde{z}^{(2)}$$
$$\wedge \tilde{z}^{(1)} = \tilde{z}^{(2)} \wedge C^{(1)} \tilde{z}^{(1)} = \beta^{(2)} \wedge C^{(2)} \tilde{z}^{(2)} = \beta^{(1)} \wedge \beta^{(1)} = \beta^{(2)})$$

Приведём для наглядности эту формулу частично в графических обозначениях (ср. с рис. 2)



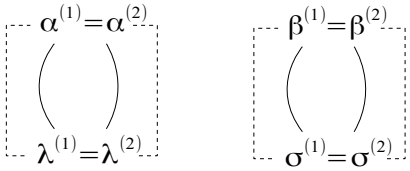
Определение

Два объекта α и β называются ***изоморфными***, если существует обратимый гребень, исходящий из α (начало гребня) и входящий в β (конец гребня).

В отличие от теории категорий начальные (конечные) гребни не всегда изоморфны друг другу.

Теорема

Два начальных объекта α и β



изоморфны тогда и только тогда, когда соединяющие их единственные гребни, соответственно, x и y допускают прямое и обратное произведение, исходящее и входящее, соответственно, в α и β.

Доказательство

Воспользуемся доказанным утверждением о совпадении гребня из начального объекта в тот же объект с этим объектом.

Условие теоремы выражается следующей формулой, с которой начат вывод

$$\exists!\, x^{(1)}\exists!\, x^{(2)}\exists!\, y^{(1)}\exists!\, y^{(2)}\exists z^{(1)}\exists z^{(2)}\exists w^{(1)}\exists u^{(1)}\exists v^{(1)}\exists w^{(2)}\exists u^{(2)}\exists v^{(2)}$$
$$\exists\tilde{z}^{(1)}\exists\tilde{z}^{(2)}\exists\tilde{w}^{(1)}\exists\tilde{u}^{(1)}\exists\tilde{v}^{(1)}\exists\tilde{w}^{(2)}\exists\tilde{u}^{(2)}\exists\tilde{v}^{(2)}$$
$$\left(\alpha^{(1)}=C^{(1)}x^{(1)}\wedge\beta^{(2)}=C^{(2)}x^{(2)}\right)\wedge\left(\beta^{(1)}=C^{(1)}y^{(1)}\wedge\alpha^{(2)}=C^{(2)}y^{(2)}\right)\wedge\alpha^{(1)}=\alpha^{(2)}\wedge\beta^{(1)}=\beta^{(2)}\wedge$$
$$\wedge\left(z^{(1)}=\mu^{(1)}x^{(1)}y^{(1)}w^{(1)}u^{(1)}v^{(1)}\wedge\alpha^{(2)}=C^{(1)}z^{(1)}\wedge z^{(2)}=\mu^{(2)}x^{(2)}y^{(2)}w^{(2)}u^{(2)}v^{(2)}\wedge\alpha^{(1)}=C^{(2)}z^{(2)}\right)\wedge$$
$$\wedge\left(\tilde{z}^{(1)}=\mu^{(1)}y^{(1)}x^{(1)}\tilde{w}^{(1)}\tilde{u}^{(1)}\tilde{v}^{(1)}\wedge\beta^{(2)}=C^{(1)}\tilde{z}^{(1)}\wedge\tilde{z}^{(2)}=\mu^{(2)}y^{(2)}x^{(2)}\tilde{w}^{(2)}\tilde{u}^{(2)}\tilde{v}^{(2)}\wedge\beta^{(1)}=C^{(2)}\tilde{z}^{(2)}\right)$$
$$\Big|$$
$$C^{(1)}\theta^{(1)}=\alpha^{(2)}\wedge\alpha^{(1)}=\alpha^{(2)}\wedge C^{(2)}\theta^{(2)}=\alpha^{(1)}\wedge C^{(1)}\tilde{\theta}^{(1)}=\beta^{(2)}\wedge\beta^{(1)}=\beta^{(2)}\wedge C^{(2)}\tilde{\theta}^{(2)}=\beta^{(1)}$$
$$\Big|$$
$$\theta^{(1)}=\lambda^{(1)}\wedge\theta^{(2)}=\lambda^{(2)}\wedge\tilde{\theta}^{(1)}=\sigma^{(1)}\wedge\tilde{\theta}^{(2)}=\sigma^{(2)}$$
$$\Big|$$
$$\exists y^{(1)}\exists y^{(2)}(\mu^{(1)}x^{(1)}y^{(1)}w^{(1)}u^{(1)}v^{(1)}=\alpha^{(2)}\wedge\mu^{(2)}x^{(2)}y^{(2)}w^{(2)}u^{(2)}v^{(2)}=\alpha^{(1)}\wedge\alpha^{(1)}=\alpha^{(2)}\wedge$$
$$\wedge\mu^{(1)}y^{(1)}x^{(1)}\tilde{w}^{(1)}\tilde{u}^{(1)}\tilde{v}^{(1)}=\beta^{(2)}\wedge\mu^{(2)}y^{(2)}x^{(2)}\tilde{w}^{(2)}\tilde{u}^{(2)}\tilde{v}^{(2)}=\beta^{(1)}\wedge\beta^{(1)}=\beta^{(2)})$$

Последняя формула задаёт обратимый гребень, необходимый для изоморфизма начальных объектов. Теорема доказана.

Мы разобрали ряд примеров теории склеек, они иллюстрируют общие теоремы, касающиеся принципа двойственности, к которому мы переходим.

Стандартная ссылка (см., например, [5] и [10]) при обсуждении принципа двойственности в категориях, как на его строгое доказательство - это книга [6]. Однако, в [6] проведено доказательство принципа двойственности, опирающееся на рассмотрение моделей. Мы проведём доказательство принципа двойственности без использования интерпретаций, основываясь на введённом понятии операций двойственности по сортам *D* и по свёрткам *B*.

Расширим набор операций $\hat{D}$ (и набор операций $\hat{B}$) перевода произвольных формул *A*, *B* сортов s_0 в формулы сортов s_1 (а также $\hat{D}^-$ и $\hat{B}^-$ для *A**, *B**) дополнительными операциями

$$\hat{D}: A\,\Sigma\,B\to\hat{D}\,A\,\Sigma\,\hat{D}\,B,\ \hat{D}:\neg A\to\neg\,\hat{D}\,A$$
$$\hat{D}^-: A^*\,\Sigma\,B^*\to\hat{D}^-\,A^*\,\Sigma\,\hat{D}^-\,B^*,\ \hat{D}^-:\neg A^*\to\neg\,\hat{D}^-\,A^*$$

, Σ любая из связок.

Для кванторов также вводим свойства $\hat{D}$ и $\hat{D}^-$:

$\hat{D}: Q\,\xi\,F\to Q\,D\,\xi\,\hat{D}\,F$, $\hat{D}^-: Q\,\xi\,F\to Q\,D^-\,\xi\,\hat{D}^-\,F$, Q - значок любого из кванторов. Те же формулы принимаем и для наборов $\hat{B}$ и $\hat{B}^-$, а также для $\hat{R}$ и $\hat{R}^-$.

Подчеркнем, что в определение *R*-копии формального языка склеек входит набор тех же аксиом, только в *R*-обозначениях. Отметим, что дуальности по сортам и по свёрткам рассматриваются отдельно, поэтому можно использовать одну и туже *R*-копию в обоих случаях.

Если склейки S и S' являются копиями друг друга, то будем писать $S \approx S'$.

Можно расширить определение дуальных по сортам и по свёрткам формул.

Определение

Пусть задана формула F исходного языка с сортами s_0, расширенного выше до формальной аксиоматической теории первого порядка (см. [9] стр. 32), тогда формула $\hat{D}F$ (соответственно, $\hat{B}F$) называется ***дуальной по сортам*** (***по свёркам***) для формулы F. Применяя к этой формуле операцию $\hat{R}^-$ мы получим формулу исходного языка, которую будем также называть ***дуальной по сортам*** (***по свёркам***) формулой.

Предложение

Пусть формула исходного языка B выведена из формул A и $A \supset B$ по правилу *modus ponens*, тогда формула $\hat{D}B$ выводится по этому правилу из формул $\hat{D}A$ и $\hat{D}(A \supset B)$. Пусть теперь формула $\forall \xi A$ получена из формулы A с помощью правила Gen классической логики. Тогда формула $\hat{D}(\forall \xi A)$ выводится из формулы $\hat{D}A$ по тому же правилу.

Доказательство (и формулировки одинаковы для обоих дуальностей, мы проводим доказательство для дуальности по сортам)

(а) Формула $\hat{D}(A \supset B)$ переводится согласно свойствам $\hat{D}$ в формулу $\hat{D}A \supset \hat{D}B$, а формула $\hat{D}(\forall \xi A)$ переводится согласно свойствам $\hat{D}$ в формулу $\forall D\xi \hat{D}A$. В расширенном языке используются правила вывода классической логики, в том числе, для формул с сортами s_1. Применяя правило *modus ponens* к $\hat{D}A$ и $\hat{D}A \supset \hat{D}B$ выводим формулу $\hat{D}B$. Применяя правило Gen к $\hat{D}A$, выводим формулу $\forall D\xi \hat{D}A$. Оба правила вывода (и только они) имеются и в интуиционисткой логике, для случая которой рассуждения также проходят. Доказательство завершено.

Предложение

Если F аксиома, то дуальная ей формула $\hat{D}F$ (соответственно, $\hat{B}F$) также является аксиомой формальной аксиоматической теории с объединенным языком склеек и его R-копии.

Доказательство

Для доказательства утверждения предложения относительно первой аксиомы отмечаем последовательно следующее (i, j нечётные целые из набора от 1 до 8)

$$C_i C_j x_k^{(1)} = C_j x_k^{(1)}; \xi_{1i} C^{(i)} \xi_{i1} \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)} = \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)};$$
$$D\left(\xi_{1i} C^{(i)} \xi_{i1} \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right) = D\left(\xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right);$$
$$\eta_{2i+1} G^{(i+1)} \eta_{i+1,2} \eta_{2j+1} G^{(j+1)} \eta_{j+1,2} y_k^{(2)} = \eta_{2j+1} G^{(j+1)} \eta_{j+1,2} y_k^{(2)};$$
$$G^{(2)} = \eta_{21} G^{(1)} \eta_{12}, G_{j+1}^{(2)} = \eta_{2j+1} G^{(j+1)} \eta_{j+1,2},$$
$$G_{i+1}^{(2)} G_{j+1}^{(2)} y_k^{(2)} = G_{j+1}^{(2)} y_k^{(2)}, \eta_{21} G_{i+1}^{(1)} \eta_{12} \eta_{21} G_{j+1}^{(1)} \eta_{12} \eta_{21} y^{(1)} = \eta_{21} G_{j+1}^{(1)} \eta_{12} \eta_{21} y^{(1)},$$
$$G_{i+1}^{(1)} G_{j+1}^{(1)} y^{(1)} = G_{j+1}^{(1)} y^{(1)}.$$

Аналогичные выкладки дают тот же результат для второй части первой аксиомы. Для свёрточной дуальности имеем при *i, j* =1,2,5,6

$$C_i C_j x_k^{(1)} = C_j x_k^{(1)}; \xi_{1i} C^{(i)} \xi_{i1} \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)} = \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)};$$
$$B\left(\xi_{1i} C^{(i)} \xi_{i1} \xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right) = B\left(\xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right);$$
$$B \xi_{1i} B^{-} B C^{(i)} B^{-} B \xi_{i1} B^{-} B \xi_{1j} B^{-} B C^{(j)} B^{-} B \xi_{j1} x_k^{(1)} = B\left(\xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right);$$
$$\eta_{3i+2} G^{(i+2)} \eta_{i+2,3} \eta_{3j+2} G^{(j+2)} \eta_{j+2,3} y_k^{(3)} = B\left(\xi_{1j} C^{(j)} \xi_{j1} x_k^{(1)}\right);$$
$$G^{(3)} G^{(3)} y_k^{(3)} = G^{(3)} y_k^{(3)}; G_3 G_3 = G_3.$$

Аналогичным образом можно получить требуемые равенства для первой аксиомы по остальным индексам, помимо 3.

Мы получили то, что требуется в предложении для нечётных чисел *i, j*. Аналогичные выкладки дают тот же результат для первой аксиомы.

Для второй аксиомы

$$\forall x_1^{(1)} \forall x_2^{(1)} \forall w'^{(1)} \forall u'^{(1)} \forall v'^{(1)} \forall x_1^{(2)} \forall x_2^{(2)} \forall w'^{(2)} \forall u'^{(2)} \forall v'^{(2)}$$
$$x^{(1)} = x^{(2)} \wedge y^{(1)} = y^{(2)} \wedge w'^{(1)} = w'^{(2)} \wedge u'^{(1)} = u'^{(2)} \wedge v'^{(1)} = v'^{(2)} \wedge$$
$$\wedge \exists x_3^{(1)} \exists x_3^{(2)} \left(\lambda^{(1)} u'^{(1)} x_1^{(1)} w'^{(1)} x_2^{(1)} v'^{(1)} = z^{(1)} \wedge \lambda^{(2)} u'^{(2)} x_1^{(2)} w'^{(2)} x_2^{(2)} v'^{(2)} = x_3^{(2)}\right) \equiv$$
$$\equiv C'^{(2)} u'^{(2)} = C^{(1)} x_1^{(1)} \wedge C'^{(1)} w'^{(1)} = C^{(2)} x_1^{(2)} \wedge$$
$$\wedge C'^{(2)} w'^{(2)} = C^{(1)} x_2^{(1)} \wedge C'^{(1)} v'^{(1)} = C^{(2)} x_2^{(2)},$$

имеем после перехода к дуальному случаю

$$\forall y_1^{(2)} \forall y_2^{(2)} \forall q'^{(2)} \forall p'^{(2)} \forall r'^{(2)} \forall y_1^{(1)} \forall y_2^{(1)} \forall q'^{(1)} \forall p'^{(1)} \forall r'^{(1)}$$
$$y_1^{(2)} = y_1^{(1)} \wedge y_2^{(2)} = y_2^{(1)} \wedge q'^{(2)} = q'^{(1)} \wedge p'^{(2)} = p'^{(1)} \wedge r'^{(2)} = r'^{(1)} \wedge$$
$$\wedge \exists y_3^{(2)} \exists y_3^{(1)} \left(\theta^{(2)} r'^{(2)} y_2^{(2)} q'^{(2)} y^{(2)} p'^{(2)} = y_3^{(2)} \wedge \theta^{(1)} r'^{(1)} y_2^{(1)} q'^{(1)} y_1^{(1)} p'^{(1)} = y_3^{(1)}\right) \equiv$$
$$\equiv G'^{(1)} p'^{(1)} = G^{(2)} y_1^{(2)} \wedge G'^{(2)} q'^{(2)} = G^{(1)} y_1^{(1)} \wedge$$
$$\wedge G'^{(1)} q'^{(1)} = G^{(2)} y_2^{(2)} \wedge G'^{(2)} r'^{(2)} = G^{(1)} y_2^{(1)}.$$

Это выражение, переписанное эквивалентным образом с переобозначениями переменных y_1 на y_2 и наоборот, *p* на *r* и наоборот

$$\forall y_1^{(1)} \forall y_2^{(1)} \forall q'^{(1)} \forall p'^{(1)} \forall r'^{(1)} \forall y_1^{(2)} \forall y_2^{(2)} \forall q'^{(2)} \forall p'^{(2)} \forall r'^{(2)}$$
$$y_1^{(1)} = y_1^{(2)} \wedge y_2^{(1)} = y_2^{(2)} \wedge q'^{(1)} = q'^{(2)} \wedge p'^{(1)} = p'^{(2)} \wedge r'^{(1)} = r'^{(2)} \wedge$$
$$\wedge \exists y_3^{(1)} \exists y_3^{(2)} \left(\theta^{(1)} p'^{(1)} y_1^{(1)} q'^{(1)} y_2^{(1)} r'^{(1)} = y_3^{(1)} \wedge \theta^{(2)} p'^{(2)} y_1^{(2)} q'^{(2)} y_2^{(2)} r'^{(2)} = y_3^{(2)}\right) \equiv$$
$$\equiv G'^{(2)} p'^{(2)} = G^{(1)} y_1^{(1)} \wedge G'^{(1)} q'^{(1)} = G^{(2)} y_1^{(2)} \wedge$$
$$\wedge G'^{(2)} q'^{(2)} = G^{(1)} y_2^{(1)} \wedge G'^{(1)} r'^{(1)} = G^{(2)} y_2^{(2)}$$

есть 2-я аксиома в R-копии склейки.

При свёрточной дуальности 2-я аксиома переходит в выражение

$$\forall y_1^{(1)} \forall y_2^{(1)} \forall q'^{(1)} \forall p'^{(1)} \forall r'^{(1)} \forall y_1^{(2)} \forall y_2^{(2)} \forall q'^{(2)} \forall p'^{(2)} \forall r'^{(2)}$$
$$y'^{(1)}_1 = y'^{(2)}_1 \wedge y'^{(1)}_2 = y'^{(2)}_2 \wedge q^{(1)} = q^{(2)} \wedge p^{(1)} = p^{(2)} \wedge r^{(1)} = r^{(2)} \wedge$$
$$\exists y'^{(1)}_3 \exists y'^{(2)}_3 (\rho'^{(1)} p^{(1)} y'^{(1)}_1 q^{(1)} y'^{(1)}_2 r^{(1)} = y'^{(1)}_3 \wedge \rho'^{(2)} p^{(2)} y'^{(2)}_1 q^{(2)} y'^{(2)}_2 r^{(2)} = y'^{(2)}_3) \equiv$$
$$\equiv G^{(2)} p^{(2)} = G'^{(1)} y'^{(1)}_1 \wedge G^{(1)} q^{(1)} = G'^{(2)} y'^{(2)}_2 \wedge$$
$$\wedge G^{(2)} q^{(2)} = G'^{(1)} y'^{(1)}_2 \wedge G^{(1)} r^{(1)} = G'^{(2)} y'^{(2)}_2 .$$

Эта формула с учетом несложного вывода (F не содержит $y'^{(1)}_3, y'^{(2)}_3$)

$$\exists y'^{(1)}_3 \exists y'^{(2)}_3 \rho' F \vdash \exists y^{(1)}_3 \exists y^{(2)}_3 \rho F$$

приводится к аксиоме (2') в R-копии склейки.

Проверка остальных аксиом также несложная, мы ее опустили. Предложение доказано.

Теорема (принцип дуальности)

Пусть формула *A* исходного языка бинарных склеек выводима, тогда дуальная по сортам формула $\hat{D}A$ (соответственно, дуальная по свёрткам формула $\hat{B}F$) также выводима.

Доказательство (и формулировки одинаковы для обоих дуальностей, мы проводим доказательство для дуальности по сортам)

По условию теоремы задана формула *A* исходного языка и её вывод из аксиом исходного языка. Сама формула является последней в дереве вывода, пусть она получена на шаге *n* - последнем шаге вывода, на котором использваны либо правило вывода *modus ponens*, либо правило Gen. Если применялось правило *modus ponens*, то имеются две формулы B и B⊃A, на основании которых получена формула A. Тогда дуальная для *A* формула $\hat{D}A$ на основании первого из доказанных выше предложений выводится из формул $\hat{D}B$, $\hat{D}B \supset \hat{D}A$ по правилу *modus ponens.* Указанный случай применения правила Gen дает тот же результат выводимости формулы $\hat{D}A$ из формулы дуальной той формуле из которой выведена в этом случае формула *A*. Теперь мы рассмотрим формулы, из которых выведена формула A, их вывод содержит менее, чем *n* шагов. Повторяя проведенный шаг мы по заданному выводу формулы A будем приходить к аксиомам и сходного языка, но по второму из доказанных выше двух последних предложений мы в листьях дерева из дуальных формул получаем аксиомы. Таким образом, вывод формулы *A* переводится в вывод дуальной формулы $\hat{D}A$. Теорема доказана.

Следствие

Если выводима формула A в исходном языке, то ей дуальная формула в том же языке $\hat{R}\hat{D}A$ (соответственно, $\hat{R}\hat{B}A$) также выводима, ее вывод является выводом дуальной формулы $\hat{D}A$ (соответственно, $\hat{B}A$), переведённым в исходный язык операцией $\hat{R}$. .

Теорема об условиях изоморфизма начальных объектов имеет ту же формулировку для конечных объектов в бинарной склейке, доказательство её есть следствие теоремы о принципе дуальности. Тем не менее, чтобы получить явное доказательство можно операцией $\hat{D}$ построить из приведённого выше вывода доказательство дуальной формулы и потом с помощью операции $\hat{R}$ выписать искомое доказательство в исходном языке.

Выбирая очередное переименование букв языка, то есть подходящую операцию *R*, мы можем многократно переходить к дуальным склейкам. Однако, мы получим ограниченное число дуальностей.

Теорема

Повторное применение операций *D* и *B* даёт копию исходной склейки *S* (*DDS*≈*S, BBS*≈*S*), применение *D* и потом *B* даёт копию склейки, полученной применением к исходной склейке сначала *B* и потом *D* (*BDS*≈*DBS*), применение *D* или *B* к склейке, полученной применением *D* и *B*, дает копию исходной склейки (*DBDS*≈*S, BDBS*≈*S*).

Таким образом, имеется всего три различных дуальности, *D, B, BD* отметим, что *BD*-дуальность нашла применение для построения сопряженного графа нейросети в подходе С.Осовского в теории искусственных нейронных сетей [8].

Рассмотренные начальные понятия и свойства бинарных склеек остаются приемлемыми и справедливыми при замене классической логики на интуиционистскую, что нетрудно проверить, используя обсуждавшийся и цитированный выше вариант интуиционистской формальной аксиоматической системы [9], в которую вписываются конструктивные операции и аксиомы бинарных направленных склеек. Интуиционистская логика обладает важнейшими атрибутами конструктивности (доказанными, в частности, в [9]) именно,

- свойством дизъюнктивности, то есть из того, что выводимо $\phi\vee\psi$, (то есть $|\text{-}\phi\vee\psi$), следует, что выводимо или ϕ , или ψ (то есть $|\text{-}\phi$ или $|\text{-}\psi$);

и свойством экзистенсиональности, если $|\text{-}\exists x\psi(x)$, то найдется терм *t* такой, что $|\text{-}\psi(t)$.

Свойства дизъюнктивности и экзистенциальности лежат в основе конструктивной логики А.А. Маркова, в рамках которой результаты для формальной теории бинарных склеек таже проходят, но этот вопрос требует отдельного изложения.

9. Заключение

Мы начали статью с обсуждения системных вопросов, к которым частично вернёмся в заключении. В развиваемой автором категорной теории систем явно выделяется конструктивная часть теории исследуемого объекта. Это позволяет наметить важную для попыток построения искусственного сознания в искусственном интеллекте границу, отделяющую естественно переносимую в компьютер часть теории от остальных ее частей, требующих ментальных представлений. Подобная общесистемная конструктивность допускает развитие теории с помощью различных логик, в частности, логики, адекватной изучаемому объекту, а не просто взятой *ad hoc*, как это обычно делается, классической логикой. Данный системный принцип (то, что логику для теории определяет объект исследования), хотя и не сформулирован в [2], но само построение теории слов, осуществлённое в книге [2], демонстрирует его применение. Тем не менее, А.Г. Драгалин выдвигает требование выбора именно интуиционистской логики для конструктивизма: «Конструктивное направление в математике является с нашей точки зрения разновидностью интуиционизма» (см., например, [9], стр. 34), причём, это очевидный инсайт эксперта, без обоснований, как это принято среди экспертов. Еще один вариант *ad hoc* выбора логики демонстрирует также в виде инсайта эксперта известный конструктивист И.Д. Заславский, категорически утверждая [11] “Если не определена логическая система, в рамках которой мы находимся, то никакие математические рассмотрения не являются возможными... каждая новая логика должна вводиться вне математики». Обсуждая физические объекты, он строит под них симметрическую логику, и применяет уже готовую логику к объекту своих исследований, которым является «предмет изучения ... процессы счета» ([11], стр. 57). Он не задается системным вопросом: если он изучает «процессы счета», то почему для построения теории этого объекта надо брать именно уже построенную без ссылкок на этот объект симметрическую логику?

Категорная теория систем опирается на огромные области системных исследований, в первую очередь на теорию функциональных систем П.К. Анохина, отсюда возникли ее постулаты, которые порождают, в частности, и указанный вопрос.

В рамках указанного системного построения в работе изучаются направленные бинарные категорные склейки, являющиеся непосредственным обобщением обычной теории категорий. Детально исследован вопрос дуальности, при этом помимо известного феномена двойственности в категориях обнаруживается еще один вид двойственности, так называемая, свёрточная двойственность, порождающая вместе с известным вариантом двойственности в

категориях третий вариант дуальности, который находит применения в теории искусственных нейронных сетей. Развитый в статье аппарат в дальнейших работах обобщается на склеечные аналоги мультикатегорий и поликатегорий, моделирующие сетевые объекты, подобные высшим категориям, а также имеющие неспайковые виды межклеточной коммуникации. В этих работах, рассмотреный в данной статье материал используется как важный модельный пример, поскольку являяется вырожденным случаем общих категорных склеек.

Автор выражает благодарность академику РАН Гончарову С.С. и академику РАН Семенову А.Л. за полезные обсуждения вопросов логики и конструктивных подходов в математике.

Список литературы

1. Tolokonnikov G. K. Categorical model of neural networks, https://openreview.net/group?id=mathai.club/MathAI/2025/Conference#tab-accept-oral
2. Марков А.А., Нагорный Н.М. Теория алгорифмов. М.: Наука, 1984. 432 с.
3. Mesarovic M.D. and Yasuhiko Takahara, General Systems Theory: Mathematical Foundations. New York.: ACADEMIC PRESS, 1975. 285 p.
4. Garner R. Polycategories via pseudo-distributive laws // Adv. Math. 2008. Vol. 218, № 3, P. 781-827.
5. Mac Lane S. Categories for the Working Mathematician. New York.: Springer, 1998. 332 p.
6. Hatcher W.S. The logical foundations of mathematics. New York.: Perg.Pr., 1982, 320 p.
7. Толоконников Г.К. Категорные склейки, категорные системы и их применение в алгебраической биологии // Биомашсистемы. т. 5. №. 1. 2021. С. 148-235.
8. Tolokonnikov G.K. Convolution Polycategories and Categorical Splices for Modeling Neural Networks // Advances in Intelligent Systems and Computing. 2020. Vol. 938, P. 259-267.
9. Драгалин А.Г. Математический интуиционизм. Введение в теорию доказательств. М.: Наука, 1979, 256 с.
10. Goldblatt R. Topoi. The categorial analysis of logic. New York.: Springer, 1984, 551 p.
11. Заславский И.Д. Симметрическая конструктивная логика. Ереван, 1978. 282 с.

UDC 004.8, 004.89

Adaptation of the language model for mathematical texts in the semantic library

Ataeva O.M. (FRC 'Computer Science and Control' RAS),
Tuchkova N.P. (FRC 'Computer Science and Control' RAS)

The paper studies the approach of LLM adaptation for queries in the mathematical subject area. The subject area is presented as an ontology of a semantic library LibMeta, where data navigation is carried out using KG *MathSemanticLib*. The descriptions of the mathematical subject area are based on mathematical encyclopedias of the Soviet and Russian mathematical schools, and the filling of the LibMeta subject area library is carried out by integrating subject areas of specialized mathematical journals. A procedure for integrating LLM and KG *MathSemanticLib* is proposed. It is shown that as a result of this approach, LLM does not go beyond the subject area, which allows us to state a more relevant answer to the query.

Keywords: *library of subject areas, large language model, knowledge graph, automation of access to scientific information, integration of knowledge graph and language model, systems of information support of scientific research, mathematical subject area, industrial engineering, ontological design*

1. Introduction

The fantastic growth of artificial intelligence (AI) methods for communicating with information systems in natural language over the past few years has led to the fact that publications on the topic of large language models (LLM) become obsolete in a year or two. Information technology has probably never experienced such speed and competition. Users are offered various software tools, in particular ChatGPT, which promise answers and predictions on all sorts of topics. A review of LLMs [15] demonstrates the various approaches used in creating these tools, and it can be concluded that the directions laid down at the beginning of the development of AI [19] continue to be implemented on new computing platforms. The authors [15] note such directions as statistical language models, neural language models, pre-trained language models and LLMs and the problems facing developers.

A well–known problem with using LLM is the difficulty of explaining and verifying the conclusions, since the answer does not indicate the sources on the basis of which the answer was formed [13]. As a rule, such sources cannot be indicated even in principle, due to the complexity

(closed nature) of algorithms for processing large amounts of data. This is especially important when it comes to scientific knowledge, especially mathematical knowledge, which is needed in a wide range of classical and applied problems. In mathematical subject areas, it is important to rely on verified sources, to distinguish LLM hallucinations from true search results.

As a rule, for scientific fields (and mathematics, of course), it is necessary to analyze a specific collection of articles, including archived ones, full texts of which are not in the public domain. These articles need to be collected in a digital library, processed, and only then can the result come to the attention of LLM. To extract knowledge from these texts, previously not found in the public domain, it is necessary to provide them for LLM training in a new subject area, i.e. to compile a corpus of articles and describe this set semantically. This endless process is still relevant, since new subject areas appear, and new interdisciplinary studies with new terminology are added to traditional ones, which means that language models need to be adapted.

One of the solutions to improve LLM response on scientific texts is the integration of LLM and the knowledge graph (KG) of the subject area [1]. In this paper, the problem of LLM adaptation is proposed to be solved by using the KG *MathSemanticLib* of the semantic library LibMeta [2], bypassing which, LLM extracts the answer from the subject area of mathematics and its application. The result is achieved due to the fact that the KG represents structured data, relies on the ontology and thesaurus of the subject area.

The structure of the article is as follows: introduction, related works, data model of the KG *MathSemanticLib*, supervised knowledge extraction example LLM answer and conclusion.

2. Related works

The issues of joint consideration of the problems of constructing KG and LLM arose naturally, as a continuation of the ideas of providing access to knowledge as structured data. The Awesome-LLM-KG (https://github.com/RManLuo/Awesome-LLM-KG) page presents a collection of links to papers and resources about unifying LLMs and KGs. It graphically displays the advantages and disadvantages of LLMs and KGs in the context of their mutual complementarity.

The main idea is that the KG structure contributes to improving LLM reasoning, and the linguistic capabilities and generalizations of LLMs improve the understanding of the essence of knowledge in the KG. Awesome-LLM-KG also provides generalizations of research directions and applications of the results of unifying LLMs and KGs in searching, building dialog systems and AI assistants, and research methods. The synergetic nature of unifying LLMs and KGs is separately noted, which is based on the mutual enrichment of LLMs and KGs when they are combined.

In the work [17] a cyclic procedure of integration of the domain knowledge and LLM is considered, as a result of which the LLM response and the domain knowledge itself are corrected, which is closest to the idea of our research.

The authors [14] provide an overview of the weaknesses of LLM related to hallucinations. The authors see an improvement in the quality of inference in the use of KG for training LLM, but they note that this process is quite complex, since first a full KG must be constructed, and then LLM reasoning with graph constraints. The work [14] proposes a procedure for transforming KG for further traversal into LLM and for generating correct reasoning paths. The option of creating a semantic description of mathematical concepts from school to university is considered in the work [5]. Here [5] provides an overview of research related to the attempt to reflect the process of cognition of mathematical subject areas and their reflection in digital resources. This idea itself has haunted the scientific community, starting with the GDML project [10]. The research [5] uses the GloVe (https://nlp.stanford.edu/projects/glove/) algorithm on a large corpus of mathematical texts to identify the frequency of use of terms and their relationships.

The comparison is made between Wikipedia terms in French and their English translations, and the use of words from the dictionary by participants (https://osf.io/dxg2w) with different mathematical backgrounds. In this way, the GloVe algorithms were tested and a relatively good correspondence between the GloVe [18] vectors and human judgments was established. This study [6] is important for our discussions in terms of the participation of experts in assessing the results of the semantic representation of subject areas and the reflection of these representations in the processes of cognition of mathematical areas. Like the authors of [6] (https://osf.io/dxg2w) [6], we use the opinion of experts, but when creating semantic images of subject areas, we rely on classical sources such as encyclopedias and monographs.

The research [7] is devoted to the description of subject areas. The data are given on how individual examples trained on corpora of specific subject areas achieve good results. However, it is noted that this is not enough to create a general approach for different subject areas. The authors propose the KnowledgeDA tool, a unified domain language model development service that can automatically generate a domain language model by performing three steps: (i) localize domain knowledge entities in texts using an embedding-similarity approach; (ii) generate enriched samples by extracting exchangeable pairs of domain entities from two representations of both the knowledge graph and the training data; (iii) select high-quality enriched samples for fine-tuning using confidence-based scoring.

A KnowledgeDA prototype for learning language models for two domains: healthcare and software development. This example of creating text corpora by subject area is quite problematic to

extend to mathematical subject areas, since the original sources may differ radically in structure and presentation features (for example, the presence of formulas changes the process of text preprocessing).

The work [11-12] is devoted to training the SciBERT (https://github.com/allenai/scibert/) model on scientific texts, where the possibilities of improving BERT after unsupervised pretraining on a large multi-domain corpus of scientific publications are demonstrated. The BERT model architecture [6] is based on a multilayer bidirectional Transformer [20] is used. Our work presents a technology for constructing KG, starting from arrays of texts of scientific mathematical and interdisciplinary journals, to the integration of the obtained KG *MathSemanticLib* with LLM in the journal recommendation system in the environment of the semantic library LibMeta [4].

3. Data Model of the KG *MathSemanticLib*

A key word is a word or a phrase. The key words are separated by commas. The number of key words is not limited. The key words are used to improve the search quality on the Journal website.

The approach used in this work is that first an ontology and thesaurus of the subject area are built, then a KG based on the ontology, and then LLM is used for communication in the library. The data structure and ontology model of the LibMeta library for the KG *MathSemanticLib* are described in the works [3], [4], here we will note only some of their properties, namely: integration of various sources (encyclopedias, monographs, journals, classifiers, thesauri, dictionaries, formulas) based on the ontology; construction of a KG based on the ontology; use of a KG for organizing a dialogue in the library. Thesauri contain the main terms of LibMeta subject areas, linked by hierarchical and horizontal relationships. The data model in LibMeta is an ontology in OWL (which is represented as an RDF graph). Filling the library is a process of completing the ontology by integrating data in accordance with their descriptions and metadata. The subject area is defined by forming a thematic subspace in the library ontology and establishing semantic links with the basic content of the library [1-3].

The mathematical encyclopedia [8], [21], the encyclopedia of mathematical physics [9], the thesaurus of ordinary differential equations, the dictionary of special functions of mathematical physics and other Russian-language sources and components of the library [3] are used as external basic taxonomies with which publications are linked. The creation and development of the LibMeta library [4] is based on the integration of mathematical knowledge, both in the retrospective and prospective direction, by adding publications from various new subject areas of mathematics, related sciences and applications.

3.1. LibMeta Ontology

The LibMeta digital library ontology defines the data structure. The concepts that make up the LibMeta ontology are conventionally divided into concepts intended for:

– describing the content of a subject area;

– forming a thesaurus of any subject area;

– describing thematic collections;

– describing the task of integrating library content with source data from LOD.

Semantically significant connections are defined between these groups of concepts. The following formal definitions are used to describe the ontology:

Definition 1. Library thesaurus TH = {T,R}, where T are terms and R are the relationships between them.

Definition 2. Library content C = {IR, A, IO}, where IR are types of information resources, a set of attributes A{ai}, information objects {IO}.

Definition 3. Semantic labels M = {mi} of an information object are terms that are not included in the thesaurus, but are necessary for thematic division of information objects IO within the subject area.

Definition 4. Semantically significant relationships of the library P = {Pi} are the following main relationships:

P1(t, io) thesaurus term → information object;

P2(io, t) information object → thesaurus term;

P3(r, s) information resource → class of source objects, where information resource is a general definition for information objects stored in the system; thus, in fact, information objects are instances of information resources;

P4(a, sa) information resource attribute → property of source class;

P5(io, os) information object → instance of class from data source;

P6(m, io) semantic label → information object;

P7(io,m) information object → semantic label.

In fact, the concepts are divided into three categories: the first includes definitions of the concepts of the semantic library content, the second category refers to the definition of concepts necessary to support terms in the thesaurus of the subject area, and the third includes definitions necessary to define the processes of integrating the content of these resources. Based on these definitions, the main processes are described, such as, for example, integrating data from different

sources, categorization/classification, mapping different models of source data to a given subject area, constructing equivalence classes, etc. Fig. 1 shows screenshot of LibMeta ODE thesaurus.

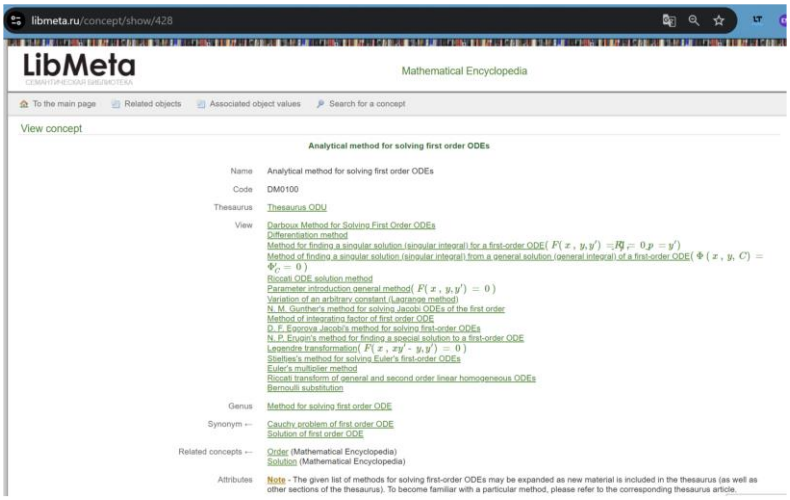


Fig. 1: An example of the ODE thesaurus concept

3.2. Completing the ontology and integrating data

The task of adding new terms (new for this ontology) and links to the ontology arises during integration with new sources (publication arrays). These are, as a rule, terms from new tasks or applications in interdisciplinary research. Integration of new data into LibMeta is implemented by completing the ontology. When integrate publications in a semantic library and KG, they must undergo preliminary processing, diagram Fig. 2.

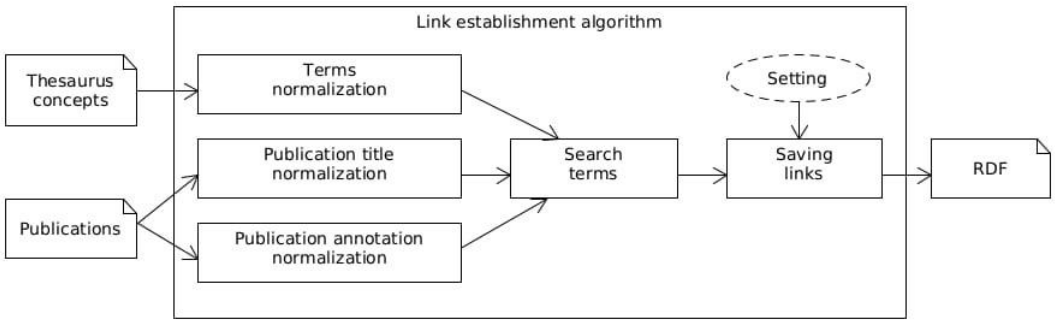


Fig. 2: Stages of publication preprocessing for ontology.

Preprocessing options depend on the source data and may vary depending on the degree of structuring of the articles. Characteristic structures are known for mathematical articles, but it is necessary to identify the main terms and links. At the Link Extraction stage (Fig. 2), semantically significant links of the library P = {Pi, i = 1, . . . , 7} are identified. If preprocessing has shown the presence of signs of belonging of the data to a certain subject area, then the publications are placed in the ontology and thesaurus of the subject area.

The task of adding new terms (new for this LibMeta ontology) and links to the ontology arises during integration with new sources (publication arrays). These are, as a rule, terms from new tasks or applications in interdisciplinary research. One of such typical examples is applications in equations of mathematical physics. Fig. 3 and Fig. 4 show the scheme and screenshot of adding terms from the journal MKMK [4], thanks to which a new subject area ’elasticity theory’, was completed and integrated into the ontology.

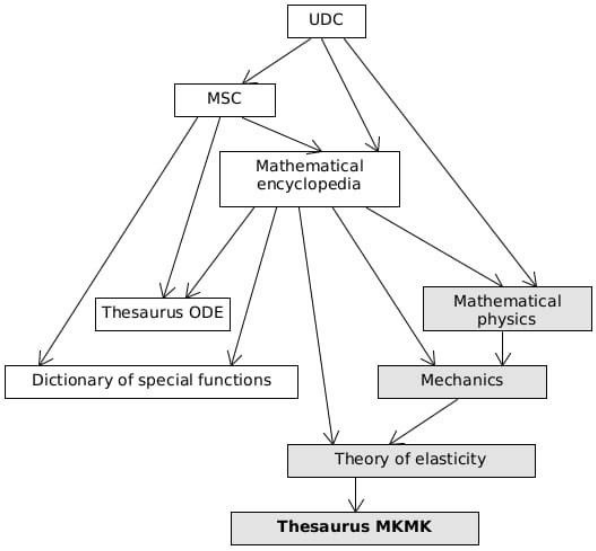


Figure 3: Connection diagram when adding a new subject area to the LibMeta ontology.

Completing the ontology, following the logic of the library data structure, affects the KG *MathSemanticLib*. Since the KG reflects the ontology connections, then when a new subject area appears in the KG of *MathSemanticLib*, a ’subgraph’ appears.

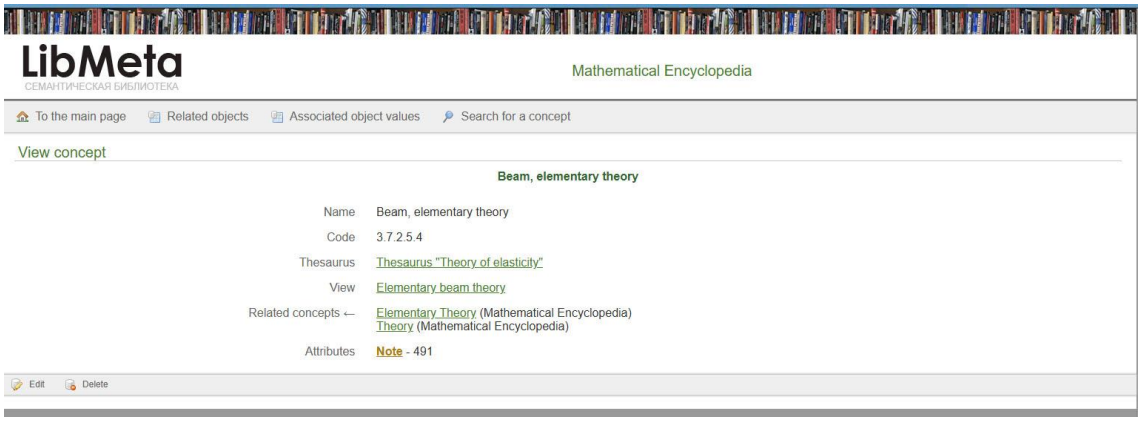


Fig. 4: Example of a thesaurus concept for ’elasticity theory’

3.3. KG *MathSemanticLib*

The LibMeta digital library ontology defines the structure of the library data. Each data element loaded into the library can be associated with an ontology node, which defines the position of the data element in the ontology. Based on the ontology links and the links defined at the design stage, a graph can be constructed. The subject area data can thus be represented as a KG, the structure of which is defined by the ontology, nodes (articles, terms, formulas) are instances of ontology elements, links are links of the subject area thesaurus. This is shown schematically as a three-level ontology in Fig. 5.

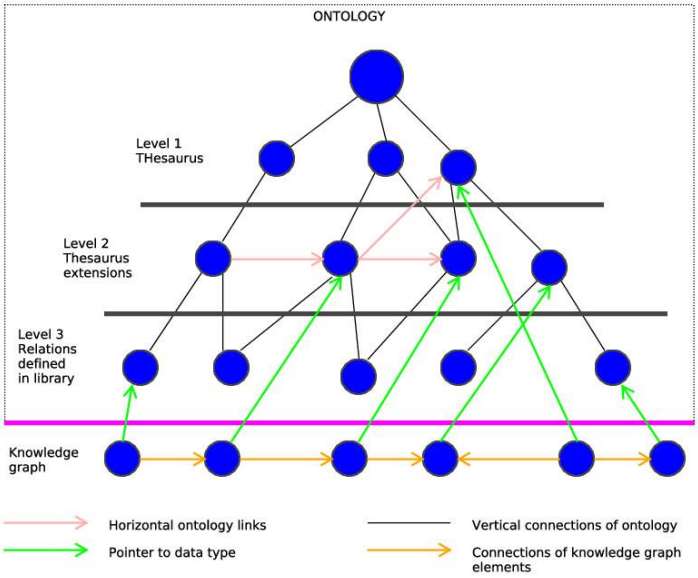


Fig. 5: Scheme of three levels of the LibMeta library ontology.

The construction of KG *MathSemanticLib* can be described in two global stages. At the first stage, a ’zero’ version of KG is constructed from some source, and at the ’second’ stage, the integration of the graph of incoming data with the general graph of the library occurs by establishing links with the thesaurus [4]. The ’zero’ version of the graph KG *MathSemanticLib* is the KG of the mathematical encyclopedia [8], [21], and the ’second’ stage is the integration of an array of scientific articles. When completing the ontology, KG is also completed, that is, the ’second’ stage is each subsequent stage.

The main stages of data processing for KG are closely related to the sources from which the data comes. Often the data is presented in an unstructured or semi-structured form. In our case, we consider, among other things, unstructured texts of Russian-language scientific articles. Nodes can be larger ontology objects Fig. 5, or objects - publication, term, person, formula.

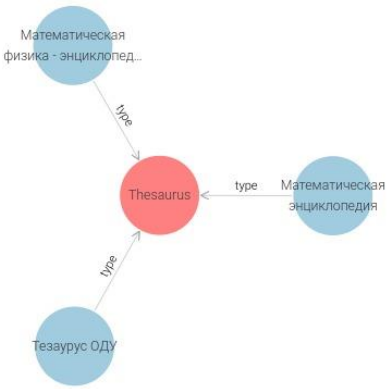


Fig. 6: The central node of the *MathSemanticLib* thesaurus.

3.4. Formulas in KG *MathSemanticLib*

The use of formula language in mathematical subject areas is a natural stage in modern dialog programs. One of the most authoritative databases of scientific publications zbMATH (https://zbmath.org) has long allowed formula entry in the search line. Nevertheless, search by formulas remains one of the tasks in information technology, since it is important to consider the meaning of their use in a scientific publication.

The LibMeta library accumulates and integrates formulas from the mathematical encyclopedia [8], [21], encyclopedia of mathematical physics [9], the thesaurus of ordinary differential equations [4], [16] dictionary of special functions and others into the ontology. The peculiarity of this integration is that the formula is saved with the context, and thus, a dictionary of formulas with links is formed, that is, the semantic image of the formula is saved. This approach allows you to include symbolic expressions in search queries. Since the formula is a semantic object in LibMeta, it can also be a KG node, for example, Fig. 6, which shows the links of the formula with other KG *MathSemanticLib* objects of the LibMeta library.

There are several types of work with formulas in the LibMeta library:

– selection of basic formulas related to concepts;

– selection of formulas related to the concept;

– selection of all formula links.

4. Supervised Knowledge Extraction Example LLM Answer

Diving into the terminology of the subject area means accepting some non-standard terms established by specialists. For mathematical subject areas, it is important to reflect the meaning of

formulas and terms when describing subject areas in digital libraries. Fig. 7–8 shows fragments of the KG *MathSemanticLib* for a concept, formula, classifiers.

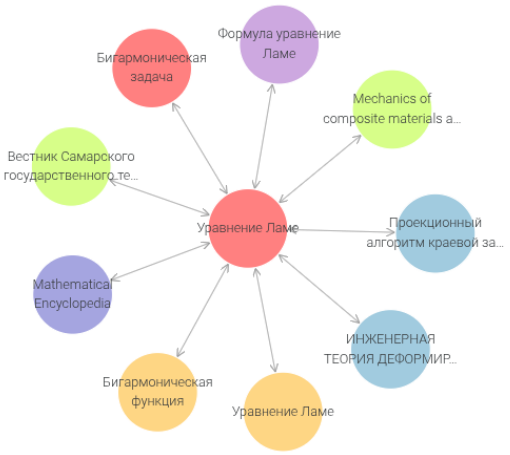


Fig. 7: Fragment of the subgraph of the concept ’Lame Equation’, KG *MathSemanticLib*.

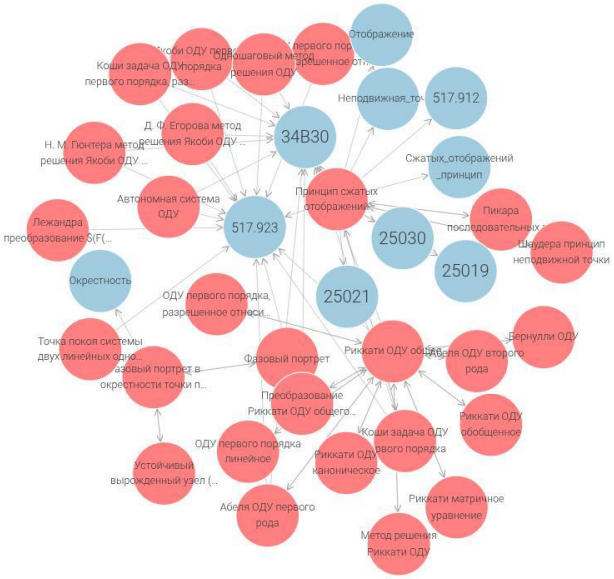


Fig. 8: Subgraph fragment with MSC codes –nodes, KG *MathSemanticLib*.

Below is a comparison of responses from some language models.

4.1. Examples with ChatGpt, YandexGPT, LLAMA

ChatGpt 4 and YandexGPT 4 Pro (https://console.yandex.cloud/) coped with the query ’Lame equation’, but gave only general information without links and formulas with an error, Perplexity and LLAMA (Large Language Model Meta AI https://app.hyperbolic.xyz/models/llama-3-3-70b-instruct) did not cope at all.

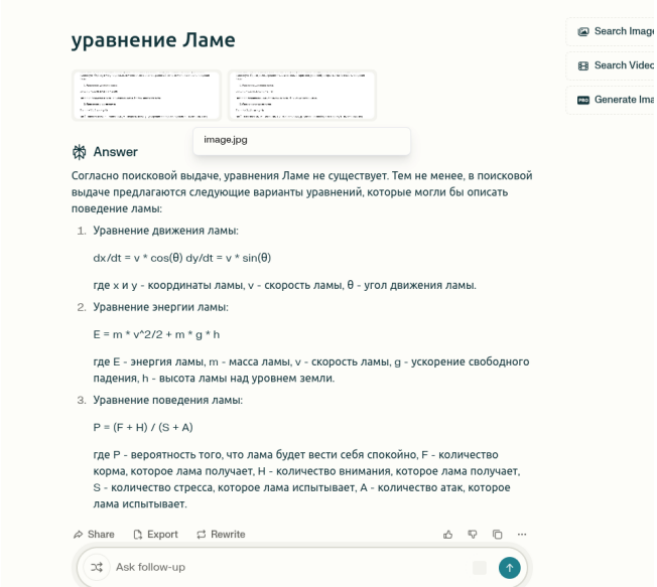


Fig. 9: Perplexity answers to the query ’Lame equation’.

The language model + knowledge graph allow using the generalizing capabilities of the language model to explore the domain graph in natural language. Questions formulated in natural language are transformed into SPARQL queries based on the ontological model of the graph. The model’s answers are given based on the mathematical encyclopedia articles and their connections Fig. 10.

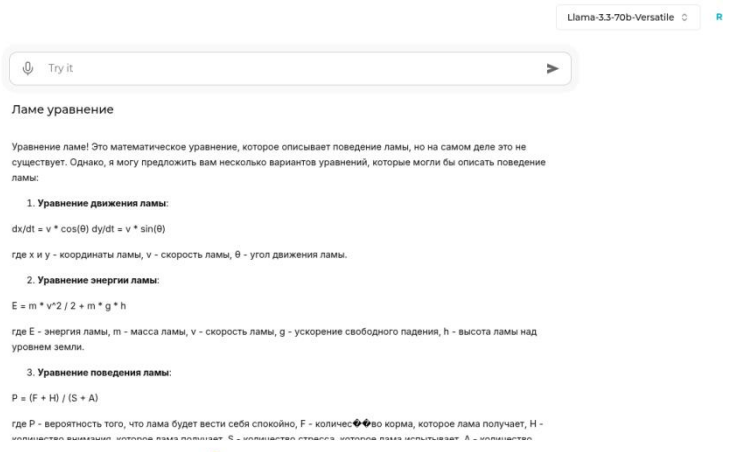


Fig. 10: LLAMA Answers to the Query ’Lame Equation’.

The model Gemma2-9b-It was used with following 3 questions. Gemma is a family of lightweight, state-of-the-art open models from Google, built from the same research and technology used to create the Gemini models. They are text-to-text, decoder-only large language models, available in English, with open weights for both pre-trained variants and instructiontuned variants. Gemma models are well-suited for a variety of text generation tasks, including question answering,

summarization, and reasoning. Their relatively small size makes it possible to deploy them in environments with limited resources such as a laptop, desktop or your own cloud infrastructure, democratizing access to state of the art AI models and helping foster innovation for everyone (https://huggingface.co/google/gemma-2-9b-it).

Question 1: How many publications?

- Generated SPARQL:

SE

LECT (COUNT(*) AS ?publications)

Question 2: How many concepts?

- SELECT (COUNT(DISTINCT ?concept) AS ?count)

WHERE

?concept a :Concept .

Question 3: How many concepts are connected with Lame formula?

- SELECT (COUNT(DISTINCT ?concept) AS ?count)

WHERE

?formula a :LameFormula .

?concept a :Concept .

?formula :connectedTo ?concept

4.2. Stages of generating a SPARQL query and response to it using a GPT like model

Stage 1. Translation of text into SPARQL query.

General format of the prompt for a model to generate a SPARQL query includes:

Question (promt): in natural language

Scheme: The graph data model is specified as an rdfs scheme, which is annotated with the rdfs:label and rdfs:comment properties, which contain a natural language description of the classes and properties of the ontology

Instructions:

- Generate a SPARQL SELECT query to query a graph database using the following ontology schema in Turtle format: {schema}
- Use only the classes and properties specified in the schema.
- Do not include any that are not explicitly provided.
- Ensure that all required prefixes are included.

- Output only the SPARQL query without any backticks or additional text.
- The question is: {prompt}

Example 1.

***Question (promt)*:** in natural laguage – How many concepts are there?

***Scheme (fragment)*:**

....

http://libmeta.ru/thesaurus/concept/DE0002

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Concept .

http://libmeta.ru/thesaurus/ODU

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Thesaurus . ...

***Answer*:**

PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#

PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

SELECT (COUNT(?concept) AS ?conceptCount)

WHERE

?concept rdf:type ¡http://libmeta.ru/Concept¿ .

Stage 2. Graph query

Next comes the step of sending the generated query through the SPARQL access point and receiving the response. For the query specified in the example, the response comes in the form ’conceptCount’: ’datatype’: ’http://www.w3.org/2001/XMLSchema#integer’, ’type’: ’literal’, ’value’: ’1221’

Stage 3. Translate the answer into human readable format

General format of the prompt to translate the response into humanreadable format includes:

***Question (promt)*:** in natural language

***Scheme*:** The graph data model is specified as an rdfs scheme, which is annotated with the rdfs:label and rdfs:comment properties, which contain a natural language description of the classes and properties of the ontology

***Instructions*:**

- Create a natural language response based solely on the results of a SPARQL query.
- You are an assistant who produces clear, human-friendly answers.

- Use only the provided information to build your response.
- This information is authoritative
- Do not question or alter it with your own knowledge.
- Ensure your answer reflects an AI assistant's tone without adding any extra details.
- If no information is available, simply state that you don't know.
- Schema: {schema}
- Context: {context}
- Question: {prompt}

Example 2.

***Question (promt)*:** in natural laguage – How many concepts are there?

***Scheme (fragment)*:** ....

http://libmeta.ru/thesaurus/concept/DE0002

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Concept .

http://libmeta.ru/thesaurus/ODU

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Thesaurus. . . . Context:

{'conceptCount': {'datatype': 'http://www.w3.org/2001/XMLSchema#integer', 'type': 'literal',

'value': '1221'}}

***Answer*:**

There are 1,221 concepts in total.

Stage 4. Correction of the query

If an invalid request was generated in response to the first step, an attempt is made to correct it using the following instructions:

***Question (promt)*:** in natural laguage

***Scheme (fragment)*:** ....

http://libmeta.ru/thesaurus/concept/DE0002

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Concept .

http://libmeta.ru/thesaurus/ODU

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Thesaurus . . . . **Error:**

Error text

SPARQL

wrong query

Instruction

- The SPARQL query below

{generated sparql is invalid.

- The error is: {error message}
- Provide a corrected version of the SPARQL query without altering its logic.
- Include only the corrected query.
- No explanations or additional text.
- Do not enclose the query in backticks.
- The ontology schema in Turtle format is provided below: {schema}

libmeta.ru/Concept .

http://libmeta.ru/thesaurus/ODU

http://www.w3.org/1999/02/22-rdf-syntax-ns#type

http://libmeta.ru/Thesaurus .

Error:

400: Error MALFORMED QUERY: QName ’lbm:ThesaurusConcept’ uses an undefined prefix

SPARQL

PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#

PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

SELECT (COUNT(?concept) AS ?conceptCount)

WHERE

?concept rdf:type lbm:ThesaurusConcept .

Answer

PREFIX rdfs: http://www.w3.org/2000/01/rdf-schema#

PREFIX rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#

PREFIX lbm: http://libmeta.ru/

SELECT (COUNT(?concept) AS ?conceptCount)

WHERE

?concept rdf:type lbm:Concept .

5. Conclusion

The proposed approach to adapting LLM to a specific mathematical direction of scientific Russian-language journals has been tested in the semantic electronic library LibMeta. The experience of integrating LLM and KG *MathSemanticLib* allows us to conclude that it is possible to organize a user dialogue with the library and create a digital assistant with the functions of a reader, author, editor, and reviewer of the journal.

It should be noted that when completing the LibMeta digital library ontology, it will be possible to ensure further integration of LLM and the KG *MathSemanticLib*, which means that the capabilities of a controlled Russian-language dialogue on mathematical topics will develop.

At the same time, the ability to check the LLM output within the LibMeta content and external sources, which will continue to be integrated, will be preserved. Further research will continue in the direction of developing dialogue and recommender systems.

References

1. Ataeva O., Serebryakov V., Tuchkova N. Development of the semantic space 'Mathematics' by integrating a subspace of its applied area // Lobachevskii J. of Mathematics. 2022. Vol. 43, No. 12. P. 29-40.
2. Ataeva O., Serebryakov V., Tuchkova N., Ontological approach to a knowledge graph construction in a semantic library // Lobachevskii J. of Mathematics. 2023. Vol. 44, No. 6. P. 2229–2239. https://doi.org/10.1134/S1995080223060471
3. Ataeva O., Serebryakov V., and Tuchkova N. From Texts to Knowledge Graph in the Semantic Library LibMeta. // Lobachevskii J. of Mathematics. 2024. Vol. 45, P. 2211–2219. https://doi.org/10.1134/S1995080224602625
4. Ataeva O., Serebryakov V., and Tuchkova N. Ontology-Driven Knowledge Graph Construction in the Mathematics Semantic Library. Pattern Recognition and Image Analysis, 2024, Vol. 34, No. 3, P. 451–458. https://doi.org/10.1134/S1054661824700196
5. Dehaene S., Debray S. Mapping and modeling the semantic space of math concepts. Cognition, 254:1–8, 2025.
6. Devlin J., Chang M.-W., Lee K., and Toutanova K. BERT: Pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT. 2019.
7. Ding R., Han X., and Wang L. A Unified Knowledge Graph Augmentation Service for Boosting Domain-specific NLP Tasks. In Findings of the Association for Computational Linguistics: ACL 2023. P 353–369, Toronto, Canada. Association for Computational Linguistics. https://aclanthology.org/2023.findings-acl.24/

8. Encyclopedia of Mathematics. https://www.encyclopediaofmath.org/index.php/Main_Pag [Электронный ресурс] (дата обращения: 30.06.2025)
9. Faddeev L. Matematicheskaya fizika. Enciklopediya, Vol. 1. Bol'shaya Rossijskaya enciklopediya, 1998.
10. Ion P. d. F., Bouche T., Misra G., Onshuus A. A., Watt S. M., and Zheng L. International Mathematical Knowledge Trust IMKT: An Update On The Global Digital Mathematics Library. Proceedings of the International Congress of Mathematicians (ICM 2018), 2019, pp. 1157-1175 https://doi.org/10.1142/9789813272880_0041
11. Iz B., Lo K. and Cohan A. SciBERT: A Pretrained Language Model for Scientific Text. Conference on Empirical Methods in Natural Language Processing (2019). https://arxiv.org/abs/1903.10676
12. https://github.com/allenai/scibert/ [Электронный ресурс] (дата обращения: 30.06.2025)
13. Kaddour J., at all. Challenges and Applications of Large Language Models / Kaddour J., Harris J., Mozes M., Bradley H., Raileanu R., McHardy R. 2023. https://arxiv.org/abs/2307.10169 https://doi.org/10.48550/arXiv.2307.10169 [Электронный ресурс] (дата обращения: 30.06.2025)
14. Luo L., Zhao Z., Gong C., Haffari G., Pan S. Graph-constrained reasoning: Faithful reasoning on knowledge graphs with large language models. arXiv preprint arXiv:2410.13080, 2024 https://doi.org/10.48550/arXiv.2410.13080
15. Malinka K., Perešíni M., Firc A., Hujňák O. and Januš F. On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree? In Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2023), July 8–12, 2023, Turku, Finland. ACM, New York, NY, USA, 7 pages. https: //doi.org/10.1145/3587102.3588827
16. Moiseev E.I., Muromskij A.A., Tuchkova N.P. Tezaurus informatsionno – poiskovyy po predmetnoy oblasti «obyknovennye differentsial'nye uravneniya» [Information search thesaurus of subject area "Ordinary Differential Equations"]. M.: MAKS Press, 2005. 116p.
17. Pan S., at all. Unifying Large Language Models and Knowledge Graphs: A Roadmap // in *IEEE Transactions on Knowledge and Data Engineering.* Pan S., Luo L., Wang Y., Chen C., Wang J., Wu X., Vol. 36. №. 7. P. 3580-3599, July 2024, https://doi.org/10.1109/TKDE.2024.3352100
18. Pennington J., Socher R., Manning C. D. GloVe: Global Vectors for Word Representation. https://nlp.stanford.edu/projects/glove/ [Электронный ресурс] (дата обращения: 30.06.2025)
19. Russell S., Norvig P. Artificial Intelligence, Global Edition. A Modern Approach. 4th edition. Munich, Pearson. 2021. 1168 p.
20. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., and Polosukhin I. 2017. Attention is all you need. In NIPS
21. Vinogradov, I.M. (red.), Matematicheskaya enciklopediya (v 5 tomah) M.: Sovetskaya enciklopediya (1977—1985)

UDC 004.853

Transforming raw corporate texts into instruction dataset for fine-tuning generator within a RAG system

Eliseev V.O. (Institute of Applied Mathematics and Mechanics)

Maksimova A.Y. (Institute of Applied Mathematics and Mechanics)

Bondarenko V.I. (Donetsk State University)

We address the problem of incorporating domain-specific knowledge into large language models (LLMs) to be used in intelligent assistants systems. We propose an algorithm for building a specifically formatted instruction dataset designed to fine-tune an LLM as a generator component within a Retrieval-Augmented Generation (RAG) system. The practical aspect of our work involved constructing an instruction dataset from raw corporate texts related to Donetsk State University (DonSU). The resulting dataset contains over 25,000 input–context–output records and will be used to fine-tune an LLM for its role as the generator in DonSU's RAG-based intelligent assistant.

Keywords: Retrieval Augmented Generation, RAG, LLM, Fine-Tuning, Instruction Dataset, AI-assistant

1. Introduction

Providing students and applicants with up-to-date information is crucial for any university. With the availability of powerful large language models (LLMs) in both open-source (e.g, LLaMA) and commercial (e.g., GPT-4) formats, it has become possible to create intelligent assistants capable of performing question-answering (QA) over corporate data and automating this communication process. This approach is not constrained by the university domain and can be used wherever QA search over a knowledge base is applicable.

Although modern LLMs demonstrate impressive capabilities, they typically lack knowledge about internal processes within organizations unless such information is publicly available and included in their training data. As a result, integrating this "hidden" knowledge into LLMs remains a significant challenge for developers of intelligent assistants.

The most common approaches to incorporating domain-specific knowledge into LLMs are building Retrieval-Augmented Generation (RAG) systems [11] and fine-tuning LLMs [3]. They enabling chosen LLM to work with unseen during pretraining domain-specific data with reason-

able effectiveness. However, both approaches suffering from their key limitations, making using them separately less efficient. These limitations will be described in more detail in Section 2.

In this paper, we propose an approach that combines supervised fine-tuning with RAG. We also introduce a method for constructing a dataset that enables an LLM, after fine-tuning on it, to effectively serve as a generator within a RAG system. Traditional instruction datasets are typically constructed as collections of input-output pairs, where the input represents a user query or instruction, and the output corresponds to the expected model response. In this work, a different approach is introduced: datasets are formulated as input-context-output triples. In this configuration, the context provides external, domain-specific information, which must be interpreted by the generator model to produce a relevant and accurate response.

The practical part of our research are based on data from official Donetsk State University (DonSU) website. Our experiments involve these steps:

1. Collecting raw texts from the website;
2. Selecting relevant texts to proceed;
3. Generating synthetic classical instructional dataset using an LLM;
4. Building a vector database over the relevant texts;
5. Adding the *context* field according to the input from the built vector database to generated instructional pairs.

The source code for each step and experiment has been made publicly available; corresponding links are provided in the relevant sections.

2. Related Work

The most common approaches for incorporating domain-specific data into LLMs are Retrieval Augmented Generation (RAG), which serves as the simplest baseline to implement and supervised fine-tuning, which is typically a more complex and resource-intensive process.

The most straightforward approach for adapting LLMs to domain-specific data is fine-tuning. Previously, assistant models were fine-tuned using dialogues that were either manually annotated or extracted from external sources, such as Twitter conversations and Reddit comments. With the emergence of instruction-based training for LLMs, OpenAI researchers fine-tuned GPT-3 on manually curated instruction data, leading to the development of InstructGPT [15], the precursor to ChatGPT. However, creating manually annotated instruction datasets for training assistant models outside industrial settings remains challenging and resource-intensive.

As a result, synthetic datasets generated by powerful LLMs or constructed using the Self-Instruct approach [18] are now commonly employed for this purpose.

There are several fine-tuning techniques exists, such as fine-tuning the whole model weights, which is costly and infeasible in non-industrial settings; adapting specific model layers while freezing the rest [12]; training the bias vectors while freezing everything else [20]; adapter tuning [4]; and others. The best way to fine-tune LLM, given the available resources for implementing an assistant, is Low-Rank Adaptation (LoRA). LoRA gives qualitative results by training a relatively small number of parameters while also providing flexibility in modifying the model's behavior during runtime [5].

By tuning an LLM on a domain-specific dataset containing classical instructional input-output pairs, we make the model simply memorize the output facts and their relations to inputs. This is not an ideal solution for building a QA system for a university, because the information about almost any university frequently changes. As a result, we would either need to continuously retrain the model on up-to-date data, which is costly and resource-intensive, or rely on lately added external sources, which the model, after instruction tuning, would struggle to utilize effectively.

A RAG also is used to enrich a large language model with domain-specific knowledge. The first mention of them was introduced by the FAIR team in 2020 [11]. Authors proposed using two models for QA search: a retriever (specifically, Dense Passage Retriever) and a generator — a transformer model. As a generator authors suggested BART-large [10], a pre-trained seq2seq transformer [17] with 400M parameters.

The retriever is responsible for vectorizing domain-specific texts, which are typically split into smaller chunks. Vectorization enables the construction of an n-dimensional representation that captures the semantic meaning of the text. The concept of vector representations was first explored by Mikolov et al. [14] during the development of the word2vec model. During RAG inference, the retriever also encodes the user query in such a way that document chunks most likely to contain the answer have the highest vector similarity to the query. The user query, together with the most relevant context chunks, is then passed to the generator, which produces a human-readable text serving as the final response.

This "Naive RAG" approach has been improved by adding metadata to document chunks, using it for document ranking (e.g., by including the document's publication date, which helps assess its current relevance), and pre- and post-retrieval processing.

A major limitation of the RAG framework lies in its reliance on the retriever, which may yield imperfect results [1]. Consequently, the generator may fail to produce a relevant response from the retrieved chunks. This typically occurs when none of the chunks contain the correct answer or when they include information about concepts only weakly related to the query's semantics, leading the model to generate incorrect outputs. The most effective strategy to address such cases is to train the model to recognize when it lacks the necessary information and to produce a predefined fallback response.

3. Proposed Method

To address the aforementioned limitations, we propose a combination of fine-tuning and RAG. Specifically, we aim to train an LLM not only to answer questions using domain-specific knowledge integrated into its weights during fine-tuning, but also to learn to extract the most relevant information from the document chunks provided by the retriever.

We propose extending the classical instructional dataset, traditionally composed of input-output pairs, by introducing a third component — *context*. The context is retrieved from a vector database constructed from the texts used to generate the original instructions, by selecting the top-n chunks most relevant to the input. Providing data as input-context-output triplets within a formatted prompt — matching the format used during assistant inference — in the fine-tuning process is expected to enhance the model's extraction capabilities, which are crucial for a generator in RAG system. Using this approach, we preserve one of the key advantages of fine-tuning — the ability to control the model's alignment and produce more domain-specific answers — while retaining a major benefit of RAG: the flexibility to update the knowledge base without the need to retrain the LLM.

This idea has been analyzed by Lin et al. [13], but this work was focused on training models to serve as generators in RAG systems working with multi-domain data in English. In our work, we focus on constructing a dataset from raw corporate texts in Russian related to a single domain of specific university — DonSU.

The official website of Don State University (DonSU) was used as the source of raw texts, as it provides structured and machine-readable content. The sequence of performed steps is shown in Figure 1.

Next, in this paper, we focus on the process of constructing a dataset in the format described above. By relevant texts, we refer to the textual content of the parsed HTML pages from the

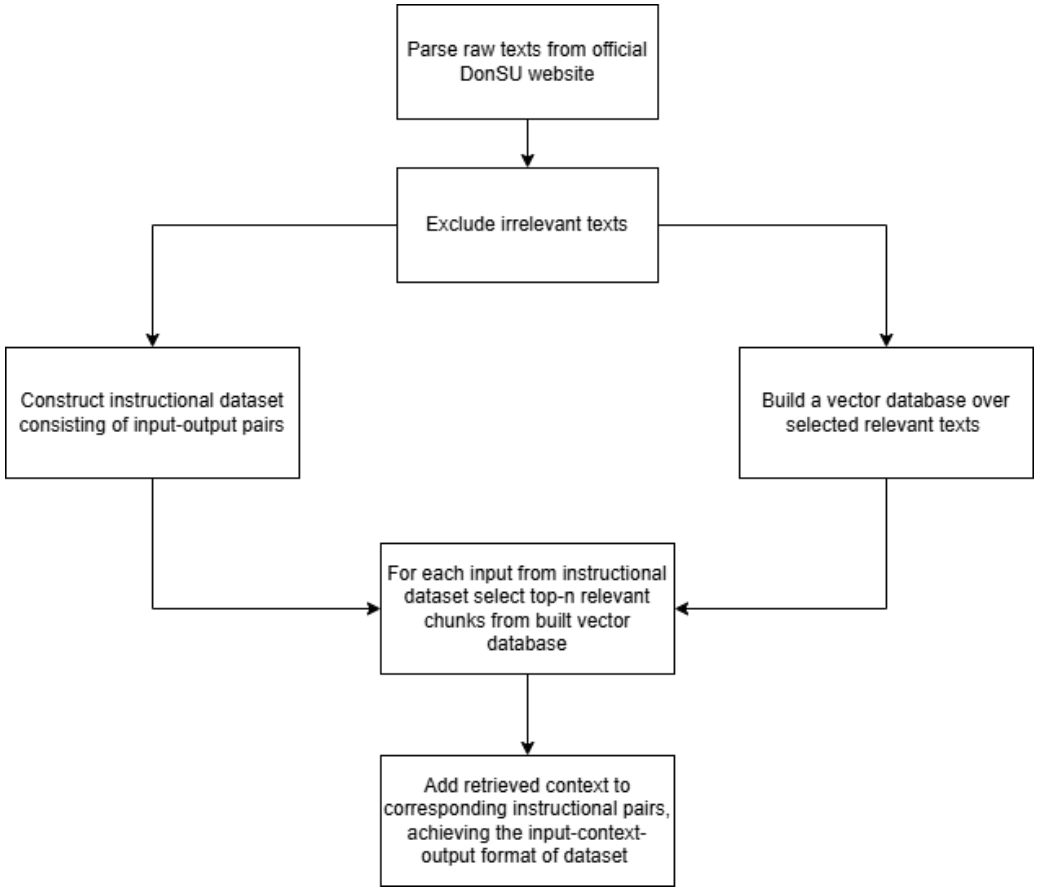


Figure 1. Performed steps during the practical part of the research.

university's website. Additionally, we downloaded all documents in textual formats (such as .pdf, .docx, .pptx, etc.), but they will only be used to populate the inference vector database.

All the steps performed, along with links to the corresponding source code, are described in detail in the following sections.

We leave the actual fine-tuning of the generator, its evaluation during RAG system inference, and the comparison of tuned models with and without the context field for future work.

4. Collecting Of Raw Texts

As the source of domain-specific data about DonSU, we chose its official website, as it provides both structured and machine-readable content.

We created a set of scripts which is open-source and available at `https://github.com/EliseevVadim/WebsiteCrawler`. The provided tools enable both parsing text data from a specified website and resulting text files processing features, including removing empty ones, merging them into a single document which is suitable for fine-tuning models via next-token prediction [2]. One of the script's arguments is a content selector, that identifies the specific

block on each HTML page on the website containing the main textual content. Putting the text content into the block with same CSS-selector is mandatory and makes the university websites machine-readable. The user can also define the depth of website traversal; if not specified, the script either crawls all available pages or stops once the defined depth is reached. Additionally, it is possible to configure whether textual files in various formats should be downloaded along with the saved HTML content. In our case, such files were downloaded, but only the texts extracted from parsed HTML pages were used in the subsequent research stages. The downloaded files were reserved for later use in populating the inference vector database.

Upon execution, the script initiates data collection from the root page of the website and recursively traverses all accessible internal hyperlinks. When encountering a URL that references an HTML page, the script extracts the main textual content and stores it in a separate file. For URLs referring to downloadable documents, the corresponding files are retrieved if this functionality has been explicitly enabled by the user. During the HTML parsing process, additional metadata — such as page titles and last modified timestamps are also collected. These metadata are expected to be beneficial for downstream tasks, including instruction generation and relevance-based ranking in the retrieval component of the vector database.

After website crawling conducted on December 11, 2024, a total of **2,912** text files were generated from the extracted content of parsed HTML pages. In addition, **3,546** documents in various formats were downloaded, comprising a total data volume of **14.3** GB. This volume of data is considered sufficient for constructing a domain-specific knowledge base to support the development of the intelligent assistant. These 2,912 files originated from HTML pages will be used in our research and will populate both RAG-powered instruction dataset and inference vector database.

5. Selecting The Relevant Texts

Analysis of the text files retrieved in the previous step revealed that some of them lacked relevant information suitable for either generating instructions or constructing the vector database for subsequent research and RAG system inference. Such files typically originated from intermediary pages containing only hyperlinks to other pages of the website or direct links to downloadable documents. Consequently, these files must be excluded from further processing.

To filter such files we decided using LLM-as-a-judge. A prompt that incorporates the content of a specific text file generated in the previous step along with its title, corresponding to the

title of the originating web page was designed. The prompt instructs an external LLM to assess the usefulness of the provided text for constructing an instruction dataset intended for LLM fine-tuning. Furthermore, a rating scale was included in the prompt to categorize each file based on its relevance.

The original prompt that was sent to LLMs was made in Russian and it is available at `https://github.com/EliseevVadim/TextsEstimator/blob/main/prompts/texts_evaluation.txt`. The English translation of the prompt is presented in Figure 2.

```
Read the text below, obtained from parsing a web page of an educational organization, and evaluate
its educational value and usefulness for training an LLM in building an instruction dataset (i.e.,
assess how well the provided text can be used to generate question-answer pairs) that will be used in
an intelligent assistant.
Consider the following criteria in your evaluation:
Content of the text.  Pay special attention to links and mentions of resources from 2022 and later,
as they are the most relevant.  Some texts are merely textual representations of links, and sometimes
entire pages contain only links.  Such files cannot be used to build a relevant instruction dataset,
so these cases should receive low scores.
File name.  The file name may indicate the topic or importance of the text and complement its
content, especially if it contains contact details.  If the name expands the context of the provided
information to a level where meaningful instructions and answers can be generated, it increases the
text's value.
Rating Scale:
1:  The text has no meaningful content, consists only of links, or serves as a placeholder without
useful information.
2:  The text contains links and headings with minimal relevant information, including outdated or
irrelevant data.
3:  The text is suitable for creating a limited number (fewer than 5) of general instructions but has
low value.
4:  The text is suitable for creating a significant number of instructions that reflect both general
and specific aspects of the organization's operations, with a sufficient share of relevant data.
5:  The text is highly structured, contains numerous relevant details (from 2022 and later), and
allows for the creation of a wide variety of valuable instructions.
Important:
The response must be a single number (from 1 to 5).  Comments, explanations, or additional information
are strictly prohibited.
File name:
***
Content:
***
```

Figure 2. Designed prompt for filtering irrelevant texts.

Next, three LLMs were selected: Gemma2-9b-it, LLaMA-3.1-70B-Instruct, and LLaMA-3.3-70B-Instruct. Prompts corresponding to all available text files were sent to each model using the NVIDIA LLM inference service using the OpenAI API. In addition to obtaining assessments for file filtering, we aimed to identify the most suitable LLM for instruction generation.

Due to input sequence length limitations imposed by some models, we restricted the evaluation to files smaller than 11 KB, a threshold determined through analysis of the file size distribution. For larger files, a default score of 5.0 was assigned, based on the assumption that longer texts are more likely to contain substantial and relevant content.

After querying the selected LLMs to evaluate the textual content extracted from the parsed HTML pages, we obtained the distributions of file evaluation scores, which are presented in Figure 3.

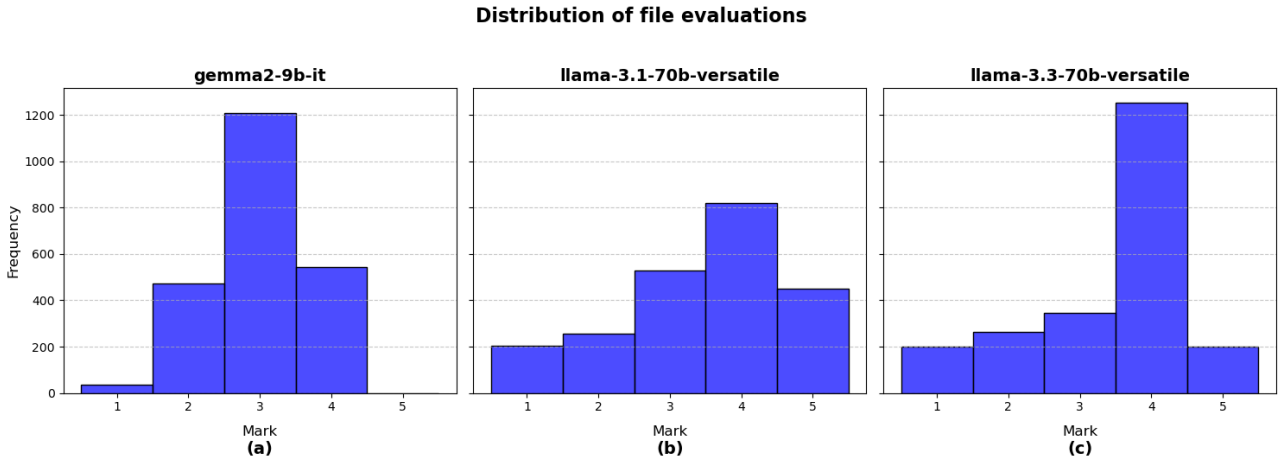


Figure 3. Distribution of file evaluations from Gemma2-9b-it (a), LLaMA-3.1-70B-Instruct (b), LLaMA-3.3-70B-Instruct (c).

As shown in the figure, all score distributions are right-skewed, with the mode occurring at the value just below the maximum. Notably, the Gemma2-9b-it model never assigned the highest score of 5.0, which may suggest its limited suitability for this type of domain-specific text evaluation. In contrast, the LLaMA-family models produced broadly similar distributions, with the most recent version, LLaMA-3.3-70B-Instruct, most frequently assigning a score of **4.0**.

Subsequently, we computed the average evaluation scores assigned by the models for each text file and analyzed the maximum divergence in scores for individual files. The greatest observed discrepancy between model evaluations was 3 points, occurring in 14 cases. Following a manual review of these instances, we concluded that the identified disagreements were not

significant enough to warrant the exclusion of the corresponding files from the datasets used for instruction generation and vector database construction.

Based on the analysis of the average score distribution across text files, we applied a thresholding criterion and excluded all files with an average score of ≤ 2.0. Such a low score typically indicates that either all models assigned a rating of 2.0 — implying that the text is nearly unusable for downstream tasks — or that the majority of models assigned a score of 1.0, marking the file as entirely irrelevant. Following this filtering step, **2,337** text files were retained, which we consider sufficient for constructing a qualitative instruction dataset. Given the assumption of approximately 10 input-output pairs per file, the resulting dataset is expected to contain over **23,000** entries. This outcome is particularly notable considering the narrow focus of our domain that contains only information related to a single university. For example, creators of RAG-Instruct reported assembling a dataset of 40,000 instruction pairs derived from the entire English Wikipedia corpus.

We additionally investigated the potential relationship between file size and the average evaluation score by constructing a correlation matrix, presented in Figure 4. As shown, the correlation between file size and average score is positive, with a value of **0.66**, indicating a moderately strong association. This finding supports the assumption that larger files are more likely to be assigned higher quality scores.

The analysis also demonstrates that models from the LLaMA family contributed most significantly to the overall average score, suggesting their higher suitability for the evaluation of raw crawled text data. Based on these results, we selected LLaMA-3.3-70B-Instruct as the primary model for instruction pair generation. The source code of the project, including the file evaluation pipeline, is open-source and available at `https://github.com/EliseevVadim/TextsEstimator`.

6. Generating Instructional Pairs

After having a list of relevant files, we proceed to construct a classical synthetic instructional dataset. In order to do this we decided using LLM prompting again. Unlike the previous section, we will utilize only one model — LLaMA-3.3-70B-Instruct, as it exhibited the best performance in evaluating the text files.

We designed a prompt that incorporates the file's title, its content, and the last update date. The model is instructed to generate the maximum number of domain-specific instruc-

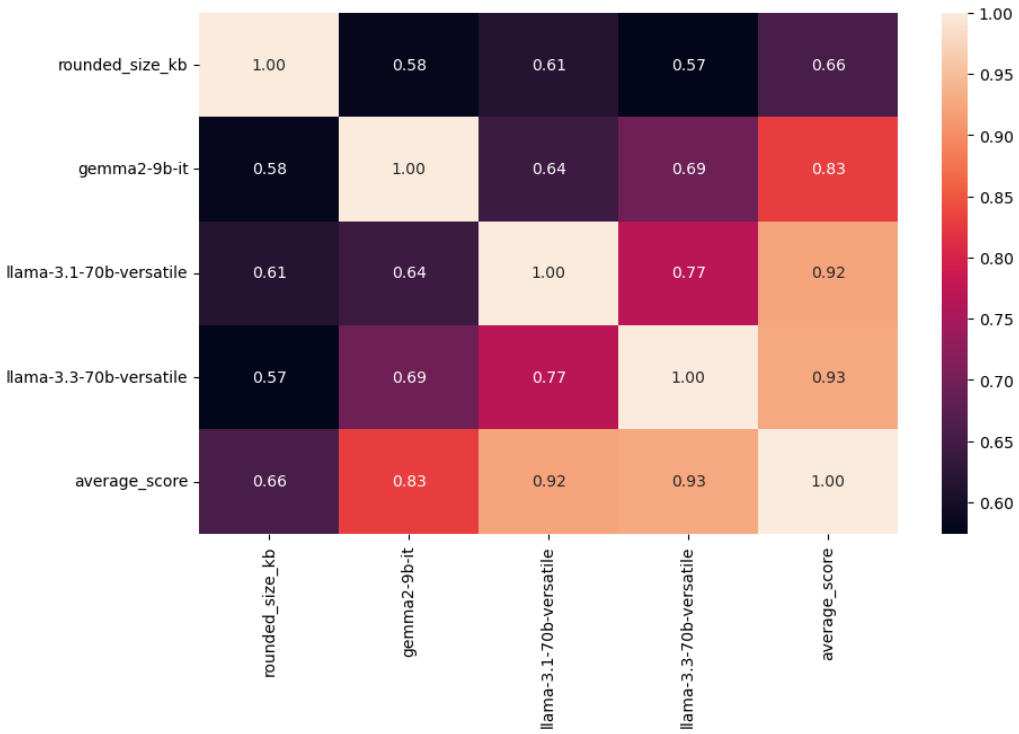


Figure 4. Correlation matrix between file size and model scores.

tional pairs, taking into account the suggested thematic directions and constraints. The original prompt also was built in Russian, it is available at `https://github.com/EliseevVadim/InstructionsGenerator/blob/main/data/prompts/instructions_creation_RU.txt`. The English translation of the prompt is presented in Figure 5.

For each relevant file, we submitted a prompt to the LLaMA-3.3-70B-Instruct model for instruction generation, following the procedure outlined in Section 4. This process resulted in the generation of 24,854 instructions, which constitutes more than half of the RAG-Instruct dataset.

Upon examination, we found that certain files, despite being deemed relevant through manual review, produced only a limited number of instructions. In light of this, we analyzed the distribution of instruction counts per file and decided to regenerate instructions for those files that initially yielded **three** or fewer instructions.

On the second pass of instruction generation, we adjusted the seed value and reduced the temperature (from 1.0 to 0.7) and top-p (from 1.0 to 0.8) parameters when calling the LLM. These lower values of temperature and top-p lead to more deterministic responses from the model, and are expected to minimize the occurrence of variances such as empty outputs or the generation of low amount of instruction pairs for files that contain enough of relevant

information.

```
Based on the HTML page data obtained from parsing the website of Donetsk State University:
Page Title:
***
Date of Last Content Update:
***
Content:
***
Generate the maximum possible number of instructions with model responses (including both long and short responses) that fully cover the provided text,
especially its factual aspects.  The instructions should be suitable for fine-tuning an LLM as an intelligent assistant for Donetsk State University.
Each instruction must be formatted as JSON with input (question) and output (answer).  Avoid generating irrelevant content that is not specifically
related to the activities of Donetsk State University.  If the text contains no meaningful information (e.g., a list of links, advertisements, or other
utility data), generate a stub with "error":  "This text does not contain useful content for generating instructions".
IMPORTANT!
The provided content often pertains to specific local aspects of the university's activities.  Therefore, DO NOT generate overly generic questions such
as "What programs are offered at DSU?" or "Who is a professor at DSU?" because the objects described in specific files are not the only ones in the
university's context, and such questions are counterproductive.
If the content mentions a faculty member, instead of a question like "Who is a professor of the Department of Physical Education and Sports on the
<title> page?" with the answer "<Name of the professor>", generate "Who is <Name of the professor>?" with the response based on the page content.
Do not respond with "A professor described on the <title> page"; instead, use the actual content of the page.  This logic applies to all other
instructions as well.
Additionally, the page title is provided, which can better reveal the text's content.  Use it and its connection to the text to construct
higher-quality instructions.
Moreover, the date of the last content update for each page is indicated.  Use this information to improve the quality of the generated instructions.
Avoid asking questions like "When was the page content last updated?" as this question is not relevant; the date should only be used to add context to
the content.
Avoid using pronouns or generic nouns when generating questions.  Instead, use named entities from the page title or as appropriate to the content.
Additionally, when listing facts in the model's response, include all available facts without truncating them with phrases like "and others."
The model being trained will serve as an intelligent assistant for an educational organization.  Therefore, this aspect must be the primary
consideration when creating instructions and responses-all must be in the context of the specific educational organization being referenced.
The text was obtained by automatically crawling the university website.  This prompt is also generated automatically by substituting data obtained
during the crawl.  Therefore, it may contain irrelevant or useless information.  For such texts, generate JSON with the error field containing the
value "This text does not contain useful content for generating instructions." Do not generate instructions for meaningless content!
Possible Types of Instructions:
1.  Questions to extract facts or details from the text (if factual material is present, questions must cover all of it).  2.  Questions requiring
analysis or comparison of information.  3.  Any other questions applicable for fine-tuning an intelligent assistant for DSU. Instructions and
responses must include both concise and detailed formats, fully covering the entire text (every part of the text MUST strictly participate as part of a
response).
Output is allowed ONLY in JSON format; any other textual content is strictly prohibited!  All output must be presented in Russian language only!
```

Figure 5. Designed prompt for generating instructions.

As a result, we generated an additional 806 instructions from 91 "problematic" files, increasing the total dataset size to **25,633** instructions. This dataset volume is deemed sufficient for the subsequent fine-tuning of the LLM. The source code of the instructions generation project is available at `https://github.com/EliseevVadim/InstructionsGenerator`.

7. Extending Instruction Dataset By Relevant Context

After successfully creating classical instruction dataset, firstly we built the vector database over relevant text files selected at Section 5. Before constructing the vector database, we needed to select an appropriate retriever model. The "best" retriever should possess an encoder-only architecture, as described by Karpukhin et al. [7], since our goal is to obtain high-quality embeddings rather than generate text output, which is the function of a decoder. Additionally, an effective retriever must be capable of processing long sequences, as emphasized by Khattab and Zaharia [8], and generating embeddings with a high dimensionality d.

Since we are working with texts in Russian and our intelligent assistant will be producing responses in Russian as well, it is preferable to use a model specifically trained on Russian texts rather than one adapted from other languages. Therefore, we selected the Giga-Embeddings-instruct model as the retriever.

The selected model, Giga-Embeddings-instruct, is designed to process contexts up to 4,096 tokens and generate embeddings with a dimensionality of d=2,048. It is based on the GigaChat-Pretrain-3B architecture, where decoder attention has been replaced with encoder attention, and incorporates Latent Attention Pooling [9] for aggregation. This model is suitable for the task at hand, as it contains only 2.5 billion parameters, ensuring efficient execution on personal devices. Despite its relatively small size, the model demonstrates high-quality vectorization and ranks second in the ruMTEB benchmark as of December 27, 2024.

Following the selection of the retriever model, the next step was to determine the chunk size and chunking strategy for document processing. Since the vector database at this stage is intended solely for the creation of a fine-tuning dataset, we opted to divide the documents into relatively small chunks of **500 tokens**, with a **50-token** overlap. This approach was selected to prevent the model's context window from becoming overloaded during fine-tuning. The tokenizer of the LLaMA-3.1-8B model was employed to define the chunk size, as this is the model slated for fine-tuning with the dataset. Furthermore, to avoid redundancy and ensure the quality of the vector database, it was essential to remove duplicate text files prior to chunking. Duplicate content could result in identical fragments being retrieved during vector search, negatively affecting the efficiency of the RAG system.

Upon removing duplicates and splitting the documents into chunks, we obtained a total of **9,935** chunks. Using them, we constructed a vector database using the FAISS framework. During the embeddings generating, we applied normalization [19] enabling the use of cosine

similarity [16] for effective similarity search over the vector database.

After constructing the vector database, we proceeded to enhance our instructional dataset, which consists of input-output pairs, by incorporating relevant context. This context was obtained through the retriever model, which conducted a similarity search within the vector database based on the input field. In this manner, we will adapt the model to the specific retriever in use and train it to extract the correct answer from the retrieved documents during fine-tuning. In this process we will provide the model with **three** documents along with the input. This limitation was imposed to prevent overloading the generator's context window, as using a greater number of documents could introduce noise and negatively affect the quality of the generated answers [6].

For each input x from our synthetic instruction dataset we retrieved three semantically closest chunks c using cosine similarity $\cos(\theta)$. It can be found as:

$$\cos(\theta) = \frac{c \cdot x}{\|c\|\|x\|} \tag{1}$$

where $c \cdot i$ is a scalar product of vectors that can be found as:

$$\mathbf{c} \cdot \mathbf{x} = \sum_{i=1}^{d} c_i x_i \tag{2}$$

and $\|c\|$, $\|x\|$ are Euclidian norms of vectors c and i that can be found as:

$$\|\mathbf{c}\| = \sqrt{\sum_{i=1}^{d} c_i^2}, \quad \|\mathbf{x}\| = \sqrt{\sum_{i=1}^{d} x_i^2} \tag{3}$$

The cosine similarity takes values in the range $[-1, 1]$, where 1 indicates that the vectors are identical, 0 corresponds to orthogonality (i.e., no semantic similarity), and -1 signifies that the vectors are diametrically opposed. Consequently, the higher the value of $\cos(\theta)$, the closer the input i to chunk c. In this work, for each input, we retrieved the three passages with the highest similarity scores. Then we added these passages to our dataset as the *context* field. The source code of extending instructional pairs with relevant context from the vector database is open-source and available at `https://github.com/EliseevVadim/InstructionsGenerator`.

8. Future Work

Following the creation of the RAG-based instruction dataset, we plan to fine-tune the generator model using the LoRA approach [5]. For this task, we selected the LLaMA-3.1-8B model,

an open-source model that combines relatively high performance with a moderate number of parameters, enabling fine-tuning process even on limited computational resources.

In addition, we intend to develop a new vector knowledge base. It will include not only the documents retrieved by the method described in this work but also text extracted from various file formats gathered through website crawling.

Given the need to store substantial volumes of vectorized textual data and to ensure regular database updates, we are considering persistent storage solutions. We intend to use pgvector for this purpose. For this moment we already developed the tool, allowing administrators managing the vector database by adding a new documents and removing irrelevant text chunks from it.

We anticipate that applying the proposed method for generating a RAG-based instruction dataset will significantly enhance the generator's performance. Moreover, the model is expected to maintain high-quality answers even as domain knowledge evolves, by effectively leveraging an easily updateable knowledge base.

9. Conclusion

In this study, we proposed a method for constructing a RAG-based instruction dataset intended for fine-tuning a generator within the RAG system of an intelligent assistant for a university. The main contribution of our work lies in the development of a system specifically trained to operate on domain-specific data related to DonSU. Unlike previous studies that primarily utilized existing datasets, we created a dataset from scratch by extracting information from the university's website and applying a sequence of preprocessing and transformation steps outlined in this paper. We described the full pipeline, from parsing raw web data to enriching a classical instruction dataset with additional context retrieved from a vector database based on semantic similarity to the input. As a result, we compiled a dataset comprising over **25,000** input-context-output triplets, ready for use in fine-tuning the generator model.

In future research, we intend to fine-tune the LLaMA-3.1-8B model on the constructed dataset, expand the vector knowledge base using the full set of collected documents, and implement a complete RAG pipeline. This system is expected to deliver high-quality, contextually relevant responses, maintaining robustness regardless of updates are made to the underlying knowledge base.

Funding The study was carried out with the financial support of the Ministry of Education and Science of the Russian Federation within the framework of the state task on the topic "Development and improvement of intelligent classification and forecasting methods for pattern recognition and modeling of information processes" FREM-2024-0001 (Registration number 1023111000141-9-1.2.1)

References

1. Barnett S. et al. Seven failure points when engineering a retrieval augmented generation system //Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI. 2024. P. 194–199.
2. Brown T. et al. Language models are few-shot learners //Advances in neural information processing systems. 2020. Vol. 33. P. 1877–1901.
3. Gururangan S. et al. Don't stop pretraining: Adapt language models to domains and tasks // Proceedings Of The 58th Annual Meeting Of The Association For Computational Linguistics. 2020. P. 8342–8360.
4. Houlsby N. et al. Parameter-efficient transfer learning for NLP //International conference on machine learning. PMLR, 2019. P. 2790–2799.
5. Hu E. J. et al. Lora: Low-rank adaptation of large language models //ICLR. 2022. Vol. 1. N. 2. P. 3.
6. Izacard G. et al. Unsupervised dense information retrieval with contrastive learning // Trans. Mach. Learn. Res. 2022.
7. Karpukhin V. et al. Dense Passage Retrieval for Open-Domain Question Answering // EMNLP (1). 2020. P. 6769–6781.
8. Khattab O., Zaharia M. Colbert: Efficient and effective passage search via contextualized late interaction over bert // Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval. 2020. P. 39–48.
9. Lee C. et al. Nv-embed: Improved techniques for training llms as generalist embedding models // Proceedings of the International Conference on Learning Representations (ICLR). 2025.
10. Lewis M. et al. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension // Annual Meeting of the Association for Computational Linguistics. 2019.
11. Lewis P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks // Advances in neural information processing systems. 2020. Vol. 33. P. 9459–9474.
12. Li X. L., Liang P. Prefix-tuning: Optimizing continuous prompts for generation // Proceedings Of The 59th Annual Meeting Of The Association For Computational Linguistics And The 11th International Joint Conference On Natural Language Processing (Volume 1: Long Papers). 2021. P. 4582–4597.
13. Lin X. V. et al. Ra-dit: Retrieval-augmented dual instruction tuning // The Twelfth International

Conference on Learning Representations. 2023.

14. Mikolov T. et al. Distributed representations of words and phrases and their compositionality // Advances in neural information processing systems. 2013. Vol. 26.
15. Ouyang L. et al. Training language models to follow instructions with human feedback //Advances in neural information processing systems. 2022. Vol. 35. P. 27730–27744.
16. Salton G., Wong A., Yang C. S. A vector space model for automatic indexing // Communications of the ACM. – 1975. Vol. 18. N. 11. P. 613–620.
17. Vaswani A. et al. Attention is all you need //Advances in neural information processing systems. 2017. Vol. 30.
18. Wang Y. et al. Self-instruct: Aligning language models with self-generated instructions // Annual Meeting of the Association for Computational Linguistics. 2022.
19. Yi J., Kim B., Chang B. Embedding Normalization: Significance Preserving Feature Normalization for Click-Through Rate Prediction // 2021 International Conference on Data Mining Workshops (ICDMW). IEEE, 2021. P. 75–84.
20. Zaken E. B., Ravfogel S., Goldberg Y. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models // Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2021. P. 1–9.

UDC 004.052.42, 004.8

Conceptual Framework for Trustworthy Artificial Intelligence: Combining Large Language Models with Formal Logic Systems*

Nechesov A.V. (The Artificial Intelligence Research Center of Novosibirsk State University)

Kondratyev D.A. (A.P. Ershov Institute of Informatics Systems SB RAS)

Sviridenko D.I. (The Artificial Intelligence Research Center of Novosibirsk State University)

Anureev I.S. (A.P. Ershov Institute of Informatics Systems SB RAS)

Garanina N.O. (A.P. Ershov Institute of Informatics Systems SB RAS)

Gumirow A.V. (Novosibirsk State University)

Gorobets I.A. (Novosibirsk State University)

Dementyeva Y.Y. (Novosibirsk State University)

*This work was supported by a grant for research centers, provided by the Ministry of Economic Development of the Russian Federation in accordance with the subsidy agreement with the Novosibirsk State University dated April 17, 2025 No. 139-15-2025-006: IGK 000000C313925P3S0002.

The paper explores the problem of building trustworthy artificial intelligence based on large language models and p-computable checkers. For this purpose we present a concept of framework for reliable verification of answers obtained by large language models (LLMs). We focus on the application of this framework to digital twin systems, particularly for smart cities, where LLMs are not yet widely used due to their resource intensity and potential for hallucination. Taking into account the fact that solution verification from a suitable set of tasks is p-computable and in most cases less complex than computing and implementing the whole task, we present a methodology that uses checkers to assess the validity of LLM-generated solutions. These checkers are implemented within the methodology of polynomial-time programming in Turing-complete languages, and guarantee a polynomial-time complexity. Our system was tested on the 2-SAT problem. This framework offers a scalable way to implement trustworthy AI systems with guaranteed polynomial complexity, ensuring error detection and preventing system hangups.

Keywords: digital twins, smart city, polynomial programming methodology, Turing-complete language, semantic programming, large language model, trustworthy AI, deductive verification, 2-SAT

1. Introduction

Today, an increasing number of users are joining the use of artificial intelligence. Artificial intelligence is being applied to a wide range of tasks and this range is growing. However, there have already been many cases where the involvement of AI, in particular large language models, has led to incorrect and even dangerous decisions. Therefore, it is important for us to obtain a trusted artificial intelligence. In this paper, we propose a concept of framework for reliable verification of decisions obtained using a large language model.

The topic of trustworthiness of artificial intelligence systems has recently received increasing attention from both AI researchers and AI systems users. It is the concept of designing and operating AI systems that are guaranteed to have the characteristics that we would normally ascribe to some agent. In this case, we usually talk about safety, security, responsibility, reliability, reproducibility, efficiency, productivity, transparency, confidentiality, fairness, ethics of its actions and results. All of this also applies to AI systems. The main challenge of trustworthy AI is to find an answer to the question, how to achieve it? And often the problem is to find a toolkit that will be used to achieve the desired result.

In the case of the symbolic approach, trust can be guaranteed by the presence of a reasonable, explainable, transparent and reliable development and operation environment based on logical and probabilistic principles. At the same time the use of machine learning technologies is

dangerous due to high uncertainty and low level of transparency and validity of the results obtained. It is especially peculiar to the technology of artificial neural networks, in particular, to the technology of large language models (LLM), when we face the "black box" effect and hallucinations.

At the moment, trusted artificial intelligence is most in demand for implementation in digital twin systems ([7], [15]). We are considering the construction of digital twins for smart cities ([14]), but we are not yet able to involve LLMs due to their unreliability and high resource intensity. However, there are still plenty of tasks that require involving LLMs to get at least inaccurate solutions for multiparametric problems that are impossible or difficult to solve using analytical methods based on available resources. We realize that it is necessary to check the correctness of these solutions. Therefore our concept is invented.

It is well known that solving NP-class problems ([17]) has a high computational complexity above polynomial. However, verifying a solution is a more simple task and often belongs to class P and requires polynomial time. This concept allows us to reduce the cost of computational resources on the side of the digital twin due to verification of solutions by a checker working in polynomial time. The checkers themselves are written within the framework of polynomial-time programming methodology in Turing-complete languages ([8]). Our methodology allows us to check whether our checker corresponds to class P. The polynomial complexity check is performed following the methodology of semantic programming, where tasks are formulated following to the task approach ([16]).

Let us note another advantage of our methodology. There are a lot of cases when implementation of task solution checker is simpler than implementation of solver for this task. Since our methodology is based on implementation of checkers instead of solvers, our approach allows reducing developer efforts in these cases.

The system, which is described in detail below, was tested on the 2-SAT problem. Scenarios were considered when the LLM gave the correct answers and when it was wrong. A finite set of checkers with partial order applied to evaluate the answers. Next, the jointness of the solver obtained using LLM and a certain checker from the set is evaluated. A solution can be decided as trusted only when the domain of the original problem coincides with the domain of the solver. The advantage of this framework is that it allows us to use trustworthy artificial intelligence systems while guaranteeing polynomial complexity. In addition, the system signals errors and hangups. The 2-SAT problem is only an illustrative example of understanding the

concept of a conceptual environment for developing trusted intelligent systems. The concept itself is applicable to other classes of tasks.

This paper is divided into the following sections. The essence of the proposed concept and its fundamental foundations are outlined in the second section. The next section describes the testing of the system. A summary of the results is presented below in the conclusion.

2. Framework

In this section we present a conceptual framework for ensuring trustworthiness of AI-based solvers (for example, LLMs).

First of all, we formalize the concepts of AI-based solver and checking environment.

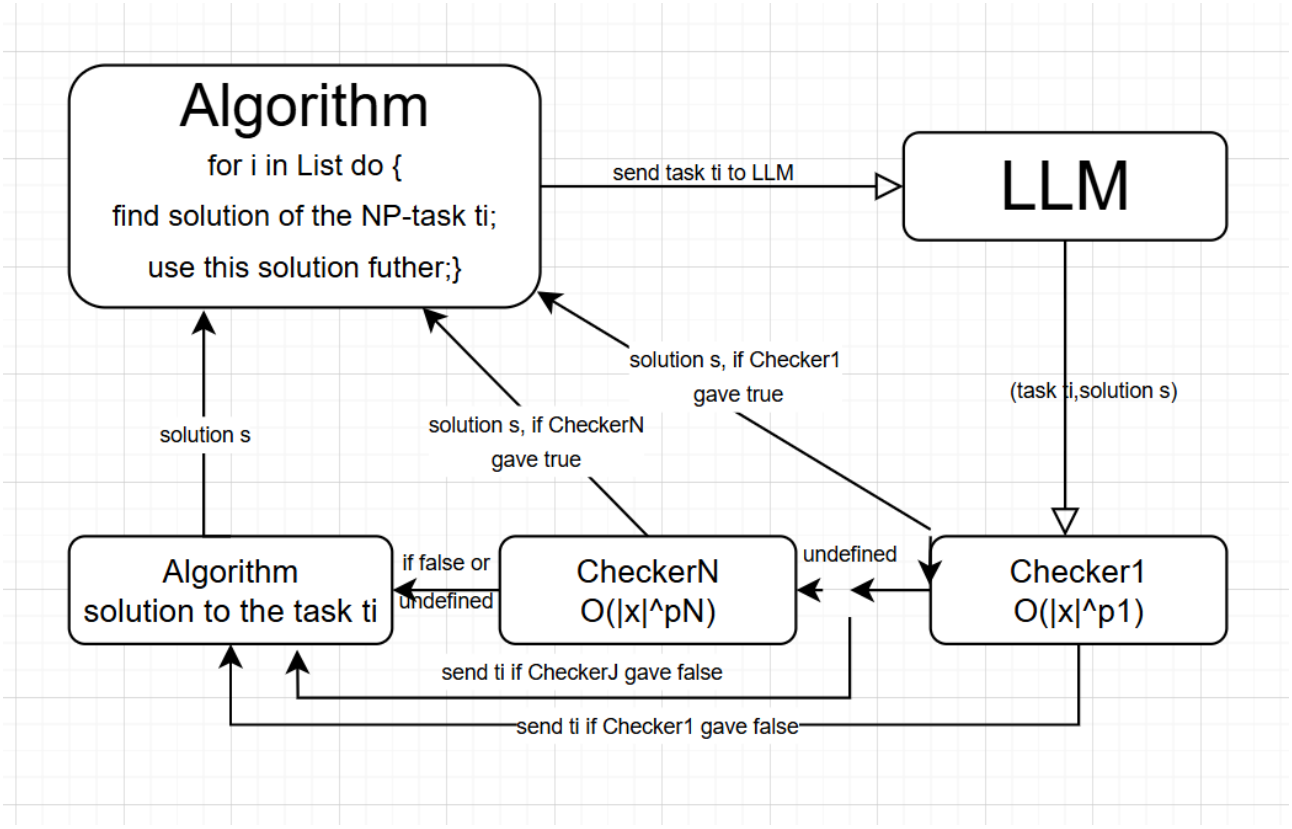


Fig. 1. The process of solving problems in combination of algorithms and LLMs

Let letters I and O (possibly with indices and primes) denote sets.

An AI-based solver S for inputs set I and outputs set O is a (possibly partial) function from $D \to R$. The partiality property means that there are inputs where the solver cannot solve the problem.

Let $bool = \{true, false\}$, and $dom(f)$ denotes the domain of a partial function f.

A checking environment E is a pair $(\Psi, \prec)$ where $\Psi = \{\psi_1, \ldots, \psi_n\}$ is a finite set of partial

functions $\psi_j \in D_j \times R_j \to bool$ called checkers, and $\prec$ is a partial order relation on Ψ such that for all $1 \le j, k \le n$ the following properties hold for each input $i \in I$:

- $j < k \wedge \psi_j \prec \psi_k \wedge i \in dom(\psi_j) \Rightarrow i \in dom(\psi_k) \wedge \psi_k(i) = \psi_j(i)$;
- $\neg(\psi_j \prec \psi_k) \wedge \neg(\psi_k \prec \psi_j) \wedge i \in dom(\psi_j) \Rightarrow$
  $i \in dom(\psi_k) \wedge \psi_k(i) = \psi_j(i) \vee i \notin dom(\psi_k)$.

The checking environment E is attached to the AI-based solver S, providing verification of the correctness of the output o of S at the input i, i.e. the correctness of the pair (i, o), by applying checkers from Ψ. The $true$ value of the checker means that the solver returned the correct result o at input i, the $false$ value does that the solver's result is incorrect at this input, and the undefined value $\perp$ does that the checker could not make an estimate.

The $\prec$ relation specifies the order in which the checkers are run.

If $\psi_j \prec \psi_k$, then the checker ψ_k can run only after the checker ψ_j returns the result, and whether the checker ψ_k is started depends on this result. The checker ψ_k is run only if the checker ψ_j failed to check the pair (i, o), i. e. ψ_j returned an undefined value $\perp$. The first property ensures that if the checker ψ_j did not fail to check the pair, then running the checker ψ_k returns the same result, that is, it makes no sense to run it.

If $\neg(\psi_j \prec \psi_k)$, and $\neg(\psi_j \prec \psi_k)$, then the checkers ψ_j and ψ_k can be run in parallel. The second property ensures consistency of the results of parallel checkers, which means that it cannot happen that one of them returned $false$ and the other returned $true$.

A solver S and an environment E are consistent if $D_j \subseteq D$, and $E_j \subseteq E$ for each $1 \le j \le n$. The consistency property means that checkers work within the state space $D \times E$ of the solver.

Now we can define trustworthiness condition for AI-based solvers.

A solver S is trustworthy w.r.t. an environment E on a set $I' \subseteq I$ if for each $i \in I'$ there exists a path $\psi_{m_1} \prec \ldots \prec \psi_{m_l}$ such that $\psi_{m_r}(i) = \perp$ for each $1 \le r \le l-1$, and $\psi_{m_l} \in bool$. The existence of the path means that checking environment estimates the solver on any inputs from I'.

A solver S is trustworthy w.r.t. an environment E if A solver S is trustworthy w.r.t. an environment E on a set I. This property means that solver S is trustworthy at any inputs.

Let us note that the system (S, E) consisting of a solver S and checking environment E can be considered as a hybrid intellectual system with intellectual part S and analytical part E that provides total or partial (on a subset of inputs) trustworthiness of S. Since checker implementation is simpler than solver implementation in many practical cases, hybrid nature

of our approach allows reducing developer efforts in practice.

There are two problems that arise when building such a hybrid AI.

First, we need to make sure that all checkers from the checking environment are working correctly. To address this problem we use formal verification methods (in particular, the deductive verification method [9]) for ensuring the correctness of the checkers themselves).

Second, we need to ensure relatively small time complexity of the checkers included in the checking environment. To address this problem we use polynomial-time programming methodology in Turing-complete languages ([8]) which allows us to check whether a checker corresponds to class P.

With this in mind, we define a conceptual framework as the quadruple $(S, E, \Delta_1, \Delta_2)$, where (S, E) has already been defined above, Δ_1 is a set of checkers verification tools, and Δ_2 is a set of checkers complexity assessment tools.

3. Experiments

In this section, we illustrate the main components of the conceptual framework using the example of solving 2-SAT problems.

3.1. Application of our framework to 2-SAT problem

2SAT is an P-class problem of assigning values to binary variables to perform a conjunction of k disjunctions. It is a special case of the general Boolean satisfiability problem. Thus, Problem of 2-SAT can be stated as: Given Conjunctive Normal Form F with each clause having only 2 terms:

$$F = (A_1 \vee B_1) \wedge (A_2 \vee B_2) \wedge (A_3 \vee B_3) \wedge \cdots \wedge (A_m \vee B_m)$$

Is it possible to assign such values to the variables so that the Conjunctive Normal Form is TRUE?

The checking environment E of our conceptual environment consists of two functions: `twosat_solver` and `sat_solution_checker`. Implementations of these functions are available in our repositories [11, 12] and also in Appendix A and Appendix B. These implementations correspond to our polynomial programming methodology (The corresponding polynomiality tests make up the component Δ_2).

The `twosat_solver` function allows us to check unsatisfiability case. Thus, $dom($`twosat_solver`$) = \{(i, o) \mid i$ is a 2-SAT task, and $o = unsat\}$. We define formal specifications of this function to

verify its implementation. This function checks the property of existence of path from x to $\neg x$ and existence of path from $\neg x$ to x in implication graph for any formula variable x. This property is important in the field of 2-SAT problem. If this property holds, than formula is unsatisfiable. Let us note that implementation of whole 2-SAT solver requires not only implementing check of this property but also implementing search of all strongly connected components of implication graph and topological sort of these components to build solution of 2-SAT problem [2]. But our approach based on using checkers in our environments allows us to avoid implementing whole 2-SAT solver. Thus, we use the `twosat_solver` function instead of implementation of whole 2-SAT solver. Consequently, we have reduced developer efforts in this case. Another function `twosat-solver` used in specifications is implementation of solution of SAT problem using generation of all combinations of possible values of formula variables. Specifications describe equivalency between this ineffective simple implementation and polynomial complex implementation of 2-SAT solution algorithm in the case of unsatisfiable formula. We have used the C-lightVer deductive verification tool [13] (an element of Δ_1 from the conceptual environment) to prove property of this equivalence described in specifications. This proof allows us to guarantee that implementation of this unsatifiablity checker is trustworthy [10].

The `sat_solution_checker` function allows us to check whether a set of variable values proposed by LLM is solution of 2-SAT problem. Thus, $dom(\texttt{sat\_solution\_checker}) = \{(i, o) \mid i \text{ is a 2-SAT task, and } o = sat\}$. The polynomial implementation of this function is based on iterations over disjunctions and variables of formula.

We use ChatGPT and DeepSeek as solvers in our conceptual framework and apply the prompt from Appendix C to solve 2-SAT problems on these LLMs.

Application of our framework to 2-SAT problem allows us to solve tasks that can be reduced to 2-SAT problem.

3.2. Heterogeneous resource allocation (HRA) in the case of two resources and one-level dependencies between tasks

We use the task of heterogeneous resource allocation (HRA) from the paper [1] as the first case study. We consider that we have two platforms, each with an unbounded number of processors. We want to execute an application represented as a Directed Acyclic Graph (DAG) using these two platforms. Each task of the application has two possible execution times, depending on the platform it is executed on. Finally, there is a cost to transfer data from one

platform to another one between successive tasks. Maximum depth of this DAG is 1. So task can depend only on task that has no dependencies. Tasks could be executed in parallel. Each task predecessors could be ran on different platforms (i.e when task 1 depends on 2 and 3, these tasks may be executed on different platforms).

The goal is to calculate minimum possible time of total DAG execution and return platform for each task.

We use a polynomial algorithm for solving this task and apply obtained polynomial algorithm to this task [1]. This algorithm is based on reduction of this task to 2-SAT problem. Reduction of considering task to 2-SAT is implemented by us and available in the repository [11]. We use application of our framework to 2-SAT problem for solving tasks obtained by this reduction.

Two examples of our first case study lead to 15 2-CNF formulas: 8 2-SAT formulas resulted from first example and 7 2-CNF formulas resulted from second example. On the one hand, all 8 2-CNF formulas corresponded to first example have been solved by both LLMs (ChatGPT and DeepSeek). Our checkers allowed us to verify solutions of these 8 2-SAT cases. On the other hand, only one formula corresponding to second example has been solved by our solver. Let us note that this formula has been solved by DeepSeek and has not been solved by ChatGPT. The set of 7 2-CNF formulas corresponded to second example contain interesting case when DeepSeek reports about unsatisfiability but our unsatisfiability checker allows us to discover that this formula is not unsatisfiable. The representation of this formula from prompt in DIMACS CNF format is follow:

```
p cnf 3 6
-3 2 0
-1 2 0
-3 -2 0
1 2 0
-1 -2 0
3 2 0
```

Let us consider DIMACS cnf format which is used by a lot of modern SAT solvers and by us to define 2-SAT instances in prompt. The number of variables and the number of clauses are defined by the line `p cnf variables clauses`. Each of line below specifies a clause: a positive literal is denoted by the corresponding number, and a negative literal is denoted by the corresponding negative number. The last number in a line should be zero.

Let us consider the classic representation of this formula:

$$(\neg x_3 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_3 \vee \neg x_2) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_3 \vee x_2)$$

This formula is not unsatisfiable (for example, this formula is true in the case of $x_1 = false$, $x_2 = true$ and $x_3 = false$).

This case is interesting since unsatisfiable case is more difficult to analyze due to absence of proposed solution in this case.

3.3. Applications in Smart Cities and Multi-Blockchains

The future of urban management is being redefined through the convergence of smart city initiatives, multi-blockchain architectures, and advanced artificial intelligence. This section presents an integrated framework where diverse blockchain systems store and process information across various domains of a smart city, smart contracts execute all business logic autonomously, and trustworthy AI underpins decision-making. Moreover, a polynomial complexity HRA-task is employed to maintain efficient control over the multi-blockchain ecosystem.

3.3.1. Architecture of the Multi-Blockchain Ecosystem

The framework implements a three-levels hierarchy:

1. **Master Blockchain (1st Level)**:
   - Root layer with maximum decentralization/security (e.g., PoW/BFT)
   - Manages cross-domain coordination and integrity proofs
2. **Sector-Specific Blockchains (2nd Level)**:
   - Domain-optimized consensus (PoS for energy, PBFT for emergency services)
   - Interfaces between subsectors and master chain
3. **Subsector Blockchains (3rd Level)**:
   - High-throughput chains for granular operations (DAG-based consensus)
   - Examples: traffic light control, household energy metering

3.3.2. Coordination and Processing Workflow

- **Bottom-Up Processing**:

$$\text{3rd level} \rightarrow \text{2nd level} \rightarrow \text{1st level}$$

  Parent chains process only after child chains complete

- **Resource Allocation**:

– GPU-intensive: PoW-like consensuses

– CPU-intensive: BFT/PoS consensuses

– Single-device execution per blockchain

3.3.3. Polynomial-Time Heterogeneous Resource Allocation (HRA) Task

Let:

$$\mathcal{B} = \{B_1, B_2, \ldots, B_n\} \text{ (blockchains)}$$
$$\mathcal{R} = \{\text{GPU}, \text{CPU}\}$$
$$T(B_i) : \text{Processing time for blockchain } B_i$$
$$B_j \prec B_k : \text{Processing order constraint}$$

Objective:

$$\min \left(\max_{r \in \mathcal{R}} \left(\sum_{B_i \text{ assigned to } r} T(B_i) \right) \right)$$

Constraints:

- Order preservation:

$$\forall B_j \prec B_k,\ \text{start\_time}(B_j) + T(B_j) \leq \text{start\_time}(B_k)$$

- Consensus-specific resource assignment:

$$\text{GPU-required } B_i \Rightarrow B_i \text{ assigned to GPU device}$$

Since the master blockchain will be processed after we have processed all other blockchains, this problem is reduced to the HRA problem for a DAG- graph with two layers. This problem can be solved in polynomial time!

The integration of multi-blockchain storage, smart contracts, trustworthy AI, and efficient resource allocation creates a resilient and adaptive infrastructure for smart cities. By partitioning data into specialized ledgers, automating business processes via smart contracts, and underpinning operations with transparent and accountable AI, urban systems can achieve unprecedented levels of efficiency, security, and scalability. Furthermore, the deployment of a polynomial complexity heterogeneous resource allocation algorithm provides the necessary control to manage diverse and dynamic resources across the entire multi-blockchain ecosystem.

3.4. Wireless sensor network (WSN) connectivity analysis

We use the task of wireless sensor network connectivity analysis from the paper [4] as the another one case study. Since wireless sensor networks (WSN) are widely applied to such perspective area as smart city creation [3], considering task is important [6].

This task is based on representation of communications between sensors as graph. This graph is referred to as communication graph. Vertexes of this graph are sensors. There is edge between sensors if and only if direct communication between these sensors exists. Let us note that this graph is directed due to possibility of only one-side direct communication in some cases (for example, when distance between two sensors does not allow sending messages from sensor with less powerful transmitter to sensor with more powerful transmitter but allows sending messages from sensor with more powerful transmitter to sensor with less powerful transmitter). Considering task is to check whether communication graph satisfies the following property: ability of each sensor to communicate with each another sensor with opportunity of using other sensors as repeaters. This property is equivalent of strong connectivity of communication graph.

The reduction of question about strong connectivity of communication graph to black-and-white 2-SAT problem has been described in the paper [4]. This reduction is interesting due its simplicity: each edge (a, b) is translated to implication $a \to b$ ($\neg a \vee b$ conjunct in obtained 2-cnf formula). But authors of the paper [4] have proved that it is necessary to avoid cases when all variables (vertexes) have *true* values in 2-SAT solution and when all variables (vertexes) have $false$ values in 2-SAT solution. Authors of the paper [4] have proposed to add the following two conjunctions to the formula to solve this problem: $x_1 \vee x_2 \vee \ldots \vee x_{n-1} \vee x_n$ and $\neg x_1 \vee \neg x_2 \vee \ldots \vee \neg x_{n-1} \vee \neg x_n$ (where $x_1, x_2, \ldots, x_{n-1}, x_n$ denotes all formula variables (all graph vertexes)). 2-SAT problem with these two additional constraints is referred to as black-and-white 2-SAT problem.

Authors of the paper [4] have proved that communication graph is strongly connected if and only if corresponding black-and-white 2-SAT problem is unsatisfiable. But it is necessary to reduce considering task to ordinary 2-SAT problem instead of black-and-white 2-SAT problem to solve obtained 2-cnf formula in polynomial time. Thus, we propose reduction of considering task to ordinary 2-sat problem. We state that two additional constraints can be replaced by statement of presence of pair of variables that have different values in obtained solution. We suggest to check this statement in the iteration over all pair of variables. We propose to add to the formula constraint $(x_i \vee x_j) \wedge (\neg x_i \vee \neg x_j)$ (where x_i and x_j are iterated variables) on

each iteration instead of adding big "black-and-white" constraints. We try to solve obtained 2-SAT problem on each iteration using application of our framework to 2-SAT problem. If obtained formula occurs satisfiable on some iteration then communication graph is not strongly connected. Else if obtained formula is unsatisfiable on each iteration then communication graph is strongly connected. Thus we improve result from the paper [4] by reducing question of graph strong connectivity to ordinary 2-SAT problem instead of black-and-white 2-SAT problem. We have implemented this reduction in our repository [12].

Let us consider example of wireless sensor network described in the paper [4]. Corresponding communication graph is presented on Figure 2.

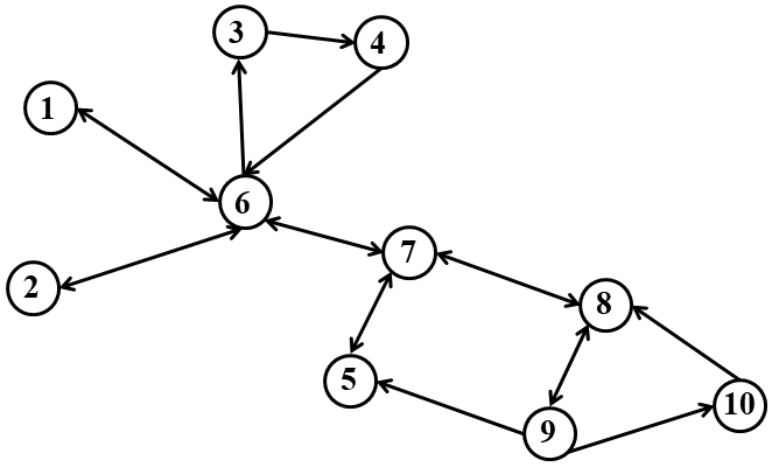


Fig. 2. Communication graph of wireless sensor network described in the paper [4]

Application of our approach to this communication graph results to 45 2-cnf formulas due to count of possible pairs of 10 variables. All of these 45 formulas are unsatisfiable. Thus, this communication graph is strongly connected. For example, let us consider the representation in DIMACS CNF format of the first formula from these 45 formulas:

```
p cnf 10 20
-1 6 0
-6 1 0
-2 6 0
-6 2 0
-6 3 0
-3 4 0
-4 6 0
-8 7 0
```

```
-6 7 0
-7 6 0
-5 7 0
-7 5 0
-7 8 0
-8 9 0
-9 8 0
-9 10 0
-10 8 0
-9 5 0
1 2 0
-1 -2 0
```

The classic representation of this formula has the following form:

$$(\neg x_1 \vee x_6) \wedge (\neg x_6 \vee x_1) \wedge (\neg x_2 \vee x_6) \wedge (\neg x_6 \vee x_2) \wedge (\neg x_6 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge$$
$$(\neg x_8 \vee x_7) \wedge (\neg x_6 \vee x_7) \wedge (\neg x_7 \vee x_6) \wedge (\neg x_5 \vee x_7) \wedge (\neg x_7 \vee x_5) \wedge (\neg x_7 \vee x_8) \wedge (\neg x_8 \vee x_9) \wedge$$
$$(\neg x_9 \vee x_8) \wedge (\neg x_9 \vee x_{10}) \wedge (\neg x_{10} \vee x_8) \wedge (\neg x_9 \vee x_5) \wedge (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$$

The first 18 conjuncts of this formula correspond to connection graph edges. The last 2 conjuncts of this formula are resulted of first iteration over variables pairs. These conjuncts state that the values of x_1 and x_2 variables are different.

Applications of solvers from our framework to this formula led to the opposite results. On the one hand, the direct answer of DeepSeek is that this formula is satisfiable. Our unsatisfiability checker shows that it is wrong answer. On the other hand, DeepSeek generates Python code to solve 2-SAT tasks. Execution of this code results to correct answer about unsatisfiability of this formula. Let us note that application of ChatGPT to this formula lead to correct answer about unsatisfiability of this formula. This case demonstrates importance of using our unsatisfiability checker.

Question of connectivity robustness of wireless sensor network relative to removing nodes or edges is important [5]. Removing direct connection from sensor 7 to sensor 6 results to modification of the example presented at Figure 3.

Application of our approach to this modified communication graph results to only 4 2-cnf formulas since solution has been found on fourth iteration. Let us consider this satisfiable formula obtained on fourth iteration:

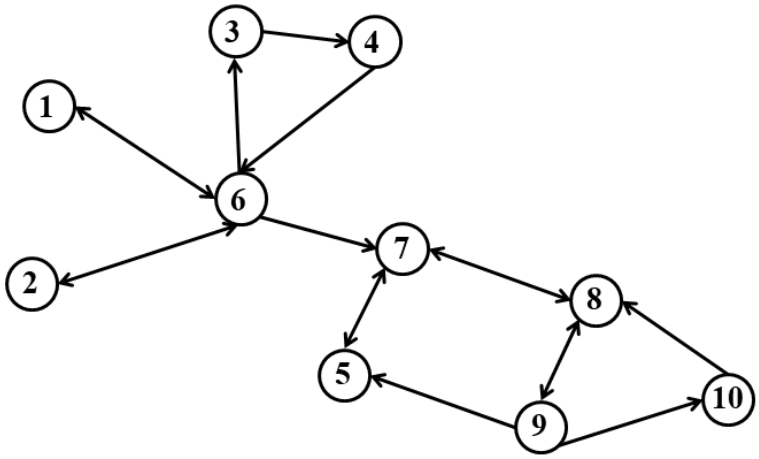


Fig. 3. Communication graph of wireless sensor network described in the paper [4] without the direct communication from sensor 7 to sensor 6

```
p cnf 10 19
-1 6 0
-6 1 0
-2 6 0
-6 2 0
-6 3 0
-3 4 0
-4 6 0
-8 7 0
-6 7 0
-5 7 0
-7 5 0
-7 8 0
-8 9 0
-9 8 0
-9 10 0
-10 8 0
-9 5 0
1 5 0
-1 -5 0
```

The classic representation of this formula has the following form:

$$(\neg x_1 \vee x_6) \wedge (\neg x_6 \vee x_1) \wedge (\neg x_2 \vee x_6) \wedge (\neg x_6 \vee x_2) \wedge (\neg x_6 \vee x_3) \wedge (\neg x_3 \vee x_4) \wedge (\neg x_4 \vee x_6) \wedge$$
$$(\neg x_8 \vee x_7) \wedge (\neg x_6 \vee x_7) \wedge (\neg x_5 \vee x_7) \wedge (\neg x_7 \vee x_5) \wedge (\neg x_7 \vee x_8) \wedge (\neg x_8 \vee x_9) \wedge (\neg x_9 \vee x_8) \wedge$$
$$(\neg x_9 \vee x_{10}) \wedge (\neg x_{10} \vee x_8) \wedge (\neg x_9 \vee x_5) \wedge (x_1 \vee x_5) \wedge (\neg x_1 \vee \neg x_5)$$

The first 17 conjuncts of this formula correspond to connection graph edges. The last 2 conjuncts of this formula are resulted from fourth iteration over variables pairs. These conjuncts state that the values of x_1 and x_5 variables are different.

Execution of Python code generated by ChatGPT for solving this task has crashed with runtime error. Thus, we have undefined result of ChatGPT solver in this case. On the one hand, the direct answer of DeepSeek is statement about satifiable of this formula. But our solution checker shows that solution proposed by DeepSeek in direct answer is incorrect. On the other hand, execution of Python code generated by DeepSeek results to correct solution. This correct solution is the following assignment: $x_1 = false, x_2 = false, x_3 = false, x_4 = false, x_5 = true, x_6 = false, x_7 = true, x_8 = true, x_9 = true, x_{10} = true$. Let us note that we do not need in using big "black-and-white" constraints to achieve this result. This case demonstrates importance of using our solution checker.

4. Conclusion

This paper presents a novel framework for the reliable verification of answers obtained by large language models (LLMs), with a focus on their application in digital twin systems for smart cities. Our experiments, conducted using the 2-SAT problem, demonstrated the effectiveness of the framework in correctly identifying trusted solutions, even in the presence of incorrect or suboptimal responses from the LLM. Let us note that advantages of reduction of tasks to 2-SAT led us to such improvement of result of the paper [4] as reduction of question about graph strong connectivity to ordinary 2-SAT problem instead of black-and-white 2-SAT problem.

The mathematical foundation of the concept, coupled with successful experimental results, provides strong evidence for the feasibility of using this framework to integrate trustworthy AI into resource-constrained environments, such as digital twins. This approach not only guarantees polynomial-time complexity but also offers a robust mechanism for error detection and system stability. Moreover our methodology can reduce developer effort due to simplicity of checker implementation relative to solver implementation.

Future work could focus on extending the framework to other problems, optimizing the verification process, and exploring its application in real-world smart city implementations.

Overall, this research contributes a significant step toward realizing reliable and efficient AI systems, facilitating their safe deployment in complex, critical environments.

References

1. Ait Aba M., Munier Kordon A., Pallez G. Scheduling on two unbounded resources with communication costs // Euro-Par 2019: Parallel Processing: 25th International Conference on Parallel and Distributed Computing, Göttingen, Germany, August 26–30, 2019, Proceedings 25. – Springer International Publishing, 2019. – pp. 117-128.
2. Aspvall B., Plass M. F., Tarjan R. E. A linear-time algorithm for testing the truth of certain quantified boolean formulas // Information processing letters. – 1979. – Т. 8. – №. 3. – pp. 121-123.
3. Belghith A., Obaidat M. S. Wireless sensor networks applications to smart homes and cities // Smart cities and homes. – Morgan Kaufmann, 2016. – pp. 17-40.
4. Biró C., Kusper G. Equivalence of strongly connected graphs and black-and-white 2-SAT problems // Miskolc Mathematical Notes. – 2018. – Vol. 19. – Is. 2. – pp. 755-768.
5. Dagdeviren O., Akram V. K. The effect of random node distribution and transmission ranges on connectivity robustness in wireless sensor networks // 2019 International Symposium on Networks, Computers and Communications (ISNCC). – IEEE, 2019. – pp. 1-5.
6. Faye S., Chaudet C. Connectivity analysis of wireless sensor networks deployments in smart cities // 2015 IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT). – IEEE, 2015. – pp. 1-6.
7. Goncharov S., Nechesov A. AI-Driven Digital Twins for Smart Cities // Engineering Proceedings. – 2023. – Vol. 58. – Is. 1. – Article ID: 94.
8. Goncharov S., Nechesov A., Sviridenko D. Programming Methodology in Turing-Complete Languages // 2024 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). – IEEE, 2024. – pp. 272-276.
9. Hähnle R., Huisman M. Deductive software verification: from pen-and-paper proofs to industrial tools // Computing and Software Science: State of the Art and Perspectives. – 2019. – pp. 345-373.
10. Kondratyev D.A., Staroletov S.M., Shoshmina I.V., Krasnenkova A.V., Ziborov K.V., Shilov N.V., Garanina N.O., Cherganov T.Y. VeHa-2024 Formal Verification Contest: Two Years of Experience and Prospects // Proceedings of the Institute for System Programming of the RAS. 2025. Volume 37. Issue 1. pp. 159–184.
11. Kondratyev D. HRA-solver: trustworthy polynomial 2SAT-based solver for heterogeneous resource allocation (HRA). 2025. URL: `https://github.com/trustworthy-code/HRA-solver` (Accessed 07 Jul 2025)
12. Kondratyev D. WSN-solver: trustworthy polynomial 2SAT-based solver for wireless sensor network (WSN) connectivity analysis. 2025. URL: `https://github.com/trustworthy-code/WSN-solver` (Accessed 07 Jul 2025)
13. Kondratyev D.A., Nepomniaschy V.A. Automation of C program deductive verification without

using loop invariants // Programming and Computer Software. 2022. Volume 48. Issue 5. pp. 331–346.

14. Nechesov A., Ruponen J. Empowering Government Efficiency Through Civic Intelligence: Merging Artificial Intelligence and Blockchain for Smart Citizen Proposals // Technologies. – 2024. – Vol. 12. – Is. 12. – Article ID. 271.
15. Nechesov A., Dorokhov I., Ruponen J. Virtual Cities: From Digital Twins to Autonomous AI Societies //IEEE Access. – 2025. – Vol. 13. – pp. 13866-13903.
16. Nechesov A.V. TASK APPROACH IN ARTIFICIAL INTELLIGENCE: LEARNING THEORY AND KNOWLEDGE HIERARCHY. 2023. Malcev meeting. Novosibirsk.
17. Prates M., Avelar P.H.C, Lemos H., Lamb L.C., Vardi M.Y. Learning to solve np-complete problems: A graph neural network for decision tsp // Proceedings of the AAAI conference on artificial intelligence. – 2019. – Vol. 33. – Is. 01. – pp. 4731-4738.

A. Appendix A

We implement checking unsatisfiability as following `twosat_solver` function written in C programming language (formal specifications of this function are written in C comments using Applicative Common Lisp language):

```
/*
(and
    (integerp
        variable_count
    )
    (<
        0
        variable_count
    )
    (integer-listp
        implication_graph_transitive_closure
    )
    (=
        (len
            implication_graph_transitive_closure
        )
        (*
            4
            (*
                variable_count
                variable_count
```

```
            )
        )
    )
)
*/
int twosat_solver(int variable_count,
    int implication_graph_transitive_closure[])
{
    int x = 0;
    int satisfiable = 1;
    /*
    (and
        (integerp
            variable_count
        )
        (<
            0
            variable_count
        )
        (integer-listp
            implication_graph_transitive_closure
        )
        (=
            (len
                implication_graph_transitive_closure
            )
            (*
                4
                (*
                    variable_count
                    variable_count
                )
            )
        )
        (integerp
            x
        )
 (<=
            0
```

```
        x
    )
    (implies
        (=
            satisfiable
            0
        )
        (and
            (=
                (nth
                    (+
                        (*
                            x
                            (*
                                variable_count
                                2
                            )
                        )
                        (+
                            x
                            variable_count
                        )
                    )
                    implication_graph_transitive_closure
                )
                1
            )
            (=
                (nth
                    (+
                        (*
                            (+
                                x
                                variable_count
                            )
                            (*
                                variable_count
                                2
                            )
```

```
                            )
                            x
                        )
                        implication_graph_transitive_closure
                    )
                    1
                )
                (<
                    x
                    variable_count
                )
            )
        )
    )
    */
    while (x < variable_count && satisfiable == 1)
    {
        if (implication_graph_transitive_closure[
              x + variable_count + x * 2 * variable_count] == 1 &&
            implication_graph_transitive_closure[
              x * 2 * variable_count + x +
                variable_count * 2 * variable_count] == 1)
        {
            satisfiable = 0;
        }
        else
        {
            x++;
        }
    }
    return satisfiable;
}
/*
(implies
    (=
        satisfiable
        0
    )
    (not
```

```
        (twosat-solver
            (boolean-variable-values
                variable_count
                nil
            )
            variable_count
            implication_graph_transitive_closure
        )
    )
)
*/
```

B. Appendix B

We implement checking solution of 2-SAT problem as following `sat_solution_checker` function written in C programming language:

```
int sat_solution_checker(int variable_count,
                         int disjunction_count,
                         int* twocnf_formula,
                         int* variable_values)
{
    int result = 1;
    for (int i = 0; (i < disjunction_count) && (result == 1);
         i++)
    {
        int index_i = 2*i;
        int first_literal = twocnf_formula[index_i];
        int second_literal = twocnf_formula[index_i + 1];

        int first_literal_variable;
        if (first_literal > 0)
        {
            first_literal_variable = first_literal;
        }
        else
        {
            first_literal_variable = -first_literal;
        }
```

```
int first_variable_value;
int first_variable_find = 0;

for (int i = 0;
     (i < variable_count) && (first_variable_find == 0);
     i++)
{
    int variable;
    if (variable_values[i] > 0)
    {
        variable = variable_values[i];
    }
    else
    {
        variable = -variable_values[i];
    }
    if (variable == first_literal_variable)
    {
        if (variable_values[i] > 0)
        {
            first_variable_value = 1;
        }
        else
        {
            first_variable_value = 0;
        }
        first_variable_find = 1;
    }
}

int second_literal_variable;
if (second_literal > 0)
{
    second_literal_variable = second_literal;
}
else
{
    second_literal_variable = -second_literal;
```

```
}

int second_variable_value;
int second_variable_find = 0;

for (int i = 0;
     (i < variable_count) && (second_variable_find == 0);
     i++)
{
    int variable;
    if (variable_values[i] > 0)
    {
        variable = variable_values[i];
    }
    else
    {
        variable = -variable_values[i];
    }
    if (variable == second_literal_variable)
    {
        if (variable_values[i] > 0)
        {
            second_variable_value = 1;
        }
        else
        {
            second_variable_value = 0;
        }
        second_variable_find = 1;
    }
}

if ((first_literal > 0) && (second_literal > 0))
{
    if ((first_variable_value == 0) &&
        (second_variable_value == 0))
    {
        result = 0;
    }
```

```
        }
        else if ((first_literal > 0) && (second_literal < 0))
        {
            if ((first_variable_value == 0) &&
                (second_variable_value == 1))
            {
                result = 0;
            }
        }
        else if ((first_literal < 0) && (second_literal > 0))
        {
            if ((first_variable_value == 1) &&
                (second_variable_value == 0))
            {
                result = 0;
            }
        }
        else
        {
            if ((first_variable_value == 1) &&
                (second_variable_value == 1))
            {
                result = 0;
            }
        }
    }
    return result;
}
```

C. Appendix C

We use the following prompt to solve 2-SAT problems on LLMs:

```
We consider 2-satisfiability (2-SAT) problem. In computer science,
2-SAT is a computational problem of assigning values to variables,
each of which has two possible values, in order to satisfy
a system of constraints on pairs of variables. It is a special
case of the general Boolean satisfiability problem. Instances of
the 2-satisfiability problem are typically expressed as Boolean
formulas of a special type, called 2-conjunctive normal form
```

```
(2-CNF) formulas. A 2-satisfiability problem may be described
using a Boolean expression with a special restricted form.
It is a conjunction (a Boolean and operation) of clauses, where
each clause is a disjunction (a Boolean or operation) of two
variables or negated variables. The variables or their negations
appearing in this formula are known as literals.

The goal is to solve a 2-SAT problem on input Boolean formula.

Use a polynomial algorithm for solving this task.

Input data is Boolean formula encoded in DIMACS CNF format.

Input data format is:
p cnf <number of variables 'N'> <number of clauses 'S'>
After that S lines of:
<first literal i where i is variable number in the case of
positive literal or i is negative variable number in the case of
negative literal> <second literal j where j is variable number
in the case of positive literal or j is negative variable number
in the case of negative literal> 0

Output data format is solution encoded in DIMACS CNF format.

Output data format is:
s <result k where k is "SATISFIABLE" in the case of satisfiable
input formula or k is "UNSATISFIABLE" in the case of
unsatisfiable input formula>
If input formula is satisfiable then representation of variable
assignments in 1 line of:
v <variable assignments a_i separated by space (where i is
variable number i from 1 to N and a_i is i in the case of
assignment of true to variable i or a_i is -i in the case of
assignment of false to variable i)> 0

Solve this task for this example:
...
```