Deductive Ergatic Design of Constructive Tasks Solutions

Anatoly P. Beltiukov, Sergey G. Maslov

Udmurt State University, Universitetskaya str., 1, 426034 Izhevsk, Russia
e-mail: belt@udsu.ru, maslov@udsu.ru

Abstract: The deductive approach to the synthesis of solutions of constructive problems with respect to ergatic systems is considered. At the same time, both the principles of deductive synthesis of solutions for ergatic systems and the processes of implementing this deductive synthesis, realized by the systems themselves, are considered.

Keywords: deductive synthesis; ergatic system; constructive realization.

1 Introduction

Providing a high quality of life, for modern people, presents them with tasks that are more and more complex. It is necessary, often, to solve these tasks with limited resources that demand the development of essentially, new, breakthrough technologies, use and organization of vital streams and processes. In this situation, modern people are forced to organize effective interaction with the natural and artificial environment of their habitat on the basis of intellectual and physical amplifiers of their capabilities, that is, to actively use the IT sphere. The key issues here are problem identification, formation and refinement of problem statements, processes of system modeling, synthesis of algorithms and programs based on description of computing environment.

In this paper, we discuss the development of systems for constructing solutions to problems by an ergatic, human-machine environment. From the point of view of the authors, it is necessary to distinguish the control problems associated with the development of a relatively simple control signal for achieving the stated goal, and the task of constructing complex objects. The complexity of the constructed objects is high enough to eliminate the solution of such a problem by looking through all the variants.

The work is carried out within the framework of the paradigm of constructive synthesis: First, we describe the capabilities of the elements of our system in a

certain domain. Then we set and formalize the problem in a language that the ergatic system understands.

Then the competent elements and subsystems build the solution of the task. If the process was successful, then the solution is a structure, which elements may refer to elements of the ergatic system. This structure can now be applied by the system in various specific circumstances (to different incoming objects). The structure can be understood and can be carried out by the elements of our ergatic system.

We set the task to provide a tool for describing what elements of the ergatic system can do in order to use these opportunities to solve the tasks assigned to the system. This work continues a series of authors' works on active layered knowledge systems [6], IT-sphere constructive directions [8], scientific reading in electronic environment [9], scientific networks [31], ergatic networks [32], ergatic systems [33], ERGANET [34], deductive synthesis [35], IT-systems design [36], ergatic intelligence [37], ergatic understanding [38].

2 Definitions

The ergative network is a human-machine information network intended for purposeful activities aimed at creating ideal and material artifacts that support the life processes of the nature-society-human system. The main purpose of the ergatic network (ERGANET) is the effective and reasonable use of knowledge to improve the quality of life of society and man.

Ergatic system is a more general concept, it is an analogue of the "organism," built on natural and artificial components. Important thing in the operation of such systems is the self-improvement of interfaces between components, structures and self-development of the entire system. The life process is the process of a stable developing existence of an object (in particular, an organism). The ergatic system, unlike the information and communication systems, is supplemented with the functions of solving constructive, creative, conative problems, i.e. tasks of constructing objects, development and motivation of subjects. Within the ergatic system itself, knowledge networks, information and communication networks can be maintained. All these components are linked by synergy and symbiosis. Knowledge in the ergatic system is inherent in active implementation (or external and internal computing). The ergatic system lays new grounds for the contexts of augmented reality (not only in visual or other sensory contexts, but also in a conceptual and metaphorical environment).

3 The Current Situation and Trends in the Creation of Information Technology Systems

At the present stage, it is possible to single out the following basic trends in the construction of information technology systems:

- Expansion of the theoretical base, in particular, the formation of a category-theoretic approach to the design of information technology systems (S. V. Kosikov [10], S. P. Kovalev [11], V. V. Antonov [12])
- Synthesis of large data and high-performance computing technologies (see e.g. [27], G. Fox [28], and also [29])
- Expansion of Autopoietic Systems (Sergeev S. F. [13]) on self-organization and self-construction of information technology systems

In this case, information and technological support systems for human activities are created, which are based on alienated knowledge and tend to replace human, arguing that by his capabilities limitations.

A new stage in the development of information technology systems is connected with human-machine systems, which are created based on synergy and symbiosis of the ideal and material, artificial and natural, as well as alienated and inalienable forms of knowledge. These are ergative systems and networks (M. Mikheev et al. [14], S. Maslov and A. Beltiukov [37] and [38]). An important feature of such systems is that in their symbiosis the integrity of a person is preserved.

Another important feature of such systems is that, despite their evolution, they also remain the integrity of man and an artificial system with divided functions of activity and non-formalized knowledge that cannot be translated into an alienated form.

The main purpose of ergatic systems and networks is to ensure the efficiency and continuity of the process of transformation "idea – abstract image – material object", effective and adequate management, as well as increasing the personal and collective threshold of difficulty to solve emerging problems. Here the principles of building systems are important, which require a revision of the basic concepts from the point of view of computing, ergaticity, attractiveness, autopoieticity, convergence (O. Rudenskiy and O. Rybak [2], S. Sergeev [15]). To this end, the following concepts have been introduced: the ergatic network (erganet), ergatic thinking, ergatic intelligence, ergatic interfaces, ergatic computing, ergatic model, ergatic statement of problems, ergatic solution and a number of others [38]. On the basis of these, the principal components of the architecture of the ergatic system are further specified.

The proposed approach to ergatical systems differs significantly from the existing approach [14] in terms of both functionality and architecture.

In the systems under consideration, the possibility of self-understanding is based on the construction of the model of itself by the system. The types of meta-understanding are: understanding of understanding and understanding of misunderstanding, understanding the boundaries of understanding, understanding and changing the environment of understanding (Metanoia [16]). See also [17].

Questions related to different approaches to the synthesis of program objects are theoretically considered in [30]: Deductive, schematic and inductive synthesis of pure solutions, performed, in particular, in interaction with a man.

Recent investigations in the area of logical-based program synthesis and transformations concern partial evaluation of programs [39], translation of an actor-based languages to the functional languages [40], verifying properties of time-aware business processes [41], synthesis software contracts from programs [42], synthesis of efficient generators for lambda terms [43], knowledge-based systems and many others. In this paper, we are primarily interested in deductive synthesis, not only of solutions, but also of the moduli of their support, performed by the ergatic system as a whole.

4 Ergatic Understanding and Ergatic Solutions of Problems

In this paper, we consider the solution of constructive problems by human-machine systems. Current consideration is limited to the tasks of creating information structures. Issues related to the creation of material objects are not considered here. In particular, we do not even consider the storage of the created structures. Thus, our constructions lie in the framework of computing. This is a generalized concept reflecting the process of changing and transforming qualitative and quantitative descriptions of systems through the processes of calculation, search, navigation, output, construction and management. A more precise mathematical definition of computing can be found in [1]. This approach is of great importance, in particular, due to the increasing spread of digital production [7].

We assert that Ergatic System is able to operate with some languages to formulate problems. The questions of goal-setting and the sources of the formulation of problems are not yet considered here. For the sake of simplicity, we assume that the common encoding problems are solved, all language objects, regardless of the nature of the languages, can be written in the form of character strings.

In order to illustrate our ideas, we consider for example the problems of constructing various information converters. It is clear that this is one of the most important problems. The solutions to problems, the objects to which they apply, and the rationale for these solutions can also be written in certain languages

available to the ergatic system. These records can have direct references to specific objects of the material environment in which the ergatic system operates. Reference can also be made to the elements and subsystems of the ergatic system itself.

The basis of ergatic constructive understanding and ergatical solution of the problems under consideration are statements of the type: "The success of applying the solution *r* of problem *B* to the object *t* is confirmed by the procedure *d*." We shall write these assertions as formulas of the form:

$$d{:}r{:}B{:}t$$

where:

B is a statement, constructively understood as a certain task

r is an object proposed as a solution to this problem

t is an object proposed for the application of this solution (for operating or testing)

d is an object for diagnosing the correctness of the object *t* for the problem *B* and for the solution *r*

These formulas will be called *verification formulas for solutions*. Naturally, to understand such formulas it is required to set the language for tasks *B*, and objects *d*, *r*, *t*. In the ergatic environment, it is required to have a human or subsystem that understands these languages. Understanding means the ability to do this with the expressions and operations described below.

Formula $d{:}r{:}B{:}t$ can be read also as: "The object *r* solves the problem *B* with counteraction *t* and support *d*." Counteraction will sometimes be called a test of the solution of the problem, because when testing a solution, it is really a test object.

From the logical point of view, the formula

$$d{:}r{:}B{:}t$$

can be considered as an abbreviation for the formula

$$(r{:}B)_{\mathrm{D}}(d;t)$$

where the colon is the realization sign according to S. C. Kleene [20] (see also A. Markov [18] and N. Shanin [19]), and *D* is the relation that uses K. Gödel in his "Dialectica" interpretation [21] (see also C. Spector [22], V. de Paiva [23], J. Avigard and S. Feferman [24], M Shirahata [25], A. Troelstra [26]). Kleene and Gödel originally proposed these relations for understanding the statements of the arithmetic of such an ideal object as an infinite set of natural numbers. Here we propose to use a similar design for understanding specific information systems. One of the main differences of the proposed system is the orientation toward the possibility of taking into account the limitations of computing resources at all

stages of the work. For example, we can restrict ourselves to procedures that can be executed in polynomial time. In this respect, this work is a continuation of the works of A. Beltiukov on intuitionistic formal theories [3], polynomially computable realizations [4], and automatic deductive synthesis [5].

Procedures (programs, instructions, etc.) in the ergatic system can be performed both by artificial executors (machines) and by natural ones (people). The executor may also be an entire ergatic subsystem. The reference to the corresponding function f in the description of the ergatic system is given in the form $f = (s,p)$, where s is an executing entity, p is a program (instruction) understood and executed by this subject s record.

$$f(x) = (s,p)(x) = s(p,x)$$

indicates the result of executing by the executor s the instruction p for the object x.

The essence of understanding the statements under consideration is that a special procedure Arb (arbitration) is defined to verify the truth of statements of the form

$$d{:}r{:}B{:}t$$

This procedure produces an appropriate truth value. That is

$$\mathrm{Arb}(d,r,B,t) \Leftrightarrow (d{:}r{:}B{:}t)$$

The execution of this procedure depends on the language system in which object (d,r,B,t) is recorded. This means that in our ergatic system for each language system L, on which object (d,r,B,t) can be described, there is an executor $S[L]$, and instruction $R[L]$ for the performance of the corresponding verification procedure by this executor:

$$\mathrm{Arb}(d,r,b,t) = S[L](R[L],d,r,B,t)$$

is the result of checking, by the executor $S[L]$ the following statement:

$$d{:}r{:}B{:}t$$

by the instruction $R[L]$. In simple cases, the test result is a classical logical truth value (for example, 0 or 1). More complex systems with multivalued logic (including fuzzy ones) are also possible.

Below is an example of how to build an instruction to verify the truth of statements of the form:

$$d{:}r{:}B{:}t$$

for a subject who understands the language of the logic of predicates (without negations and functions) on which the task B is written.

In practice, tasks can be recorded in a variety of languages, which necessitates various instructions for verifying solutions. Let's start with simple problems (i.e. atomic formulas) $B = P(\boldsymbol{c})$ with a predicate P and parameters $\boldsymbol{c}$, $\boldsymbol{c}$ are the names of

objects in the domain to which our problem relates. The corresponding instruction can be written as follows:

$$d{:}r{:}P(\boldsymbol{c}){:}t \Leftrightarrow I[P](J[P],d,r,\boldsymbol{c},t)$$

where:

r is an object used as a solution to a problem $P(\boldsymbol{c})$

t is an object used to test solution r of task $P(\boldsymbol{c})$

d is an object used to diagnose the correctness of the test t for the solution r of the task $P(\boldsymbol{c})$

(I,J) is the interpretation of predicates on the subject domain (a pair of tables, described below)

$I[P]$ is the name of the subject who understands the predicate P

$J[P]$ is the instruction for the subject $I[P]$ by understanding the predicate P

Here, for each predicate P, languages are defined for writing solutions r, tests t and diagnostic procedures d.

The understanding of complex problems (defined by means of logical connectives from simpler ones) can be determined, for example, by the following formulas (obtained by combining the corresponding constructions from [20] and [21]):

$$(p,q){:}(\mathrm{b},\mathrm{c}){:}(B\&C){:}(d,e) \Leftrightarrow (p{:}b{:}B{:}c)\&(q{:}c{:}C{:}e)$$

$$(p,q){:}(0,b){:}(B\vee C){:}(d,e) \Leftrightarrow p{:}b{:}B{:}d$$

$$(p,q){:}(1,c){:}(B\vee C){:}(d,e) \Leftrightarrow q{:}c{:}C{:}e$$

$$(g,h){:}f{:}(B\Rightarrow C){:}(a,b,c,d) \Leftrightarrow$$

$$(a{:}b{:}B{:}g(a,b,c,d)\Rightarrow h(a,b){:}f(b){:}C{:}c)$$

$$g{:}f{:}\forall xB(x){:}(c,b) \Leftrightarrow g(c){:}f(c){:}B(c){:}b$$

$$p{:}(c,b){:}\exists xB(x){:}t \Leftrightarrow p{:}b{:}B(c){:}t$$

As you can see, the executor needs to be able to break pairs into elements, distinguish zeros from ones, divide formulas into immediate components, substitute the name c of the object of the domain instead of the variable x in the formula $B(x)$, apply programs (f, g, h) to their arguments (a, b, c, d). We assume that the programs are written in a language that guarantees the completion of their execution and has the ability to generate structures and program objects, mentioned in these formulas. As a rule, for practical needs, it is necessary to require certain complexity constraints on the execution of generated programs, e.g. polynomial time bounds.

If you want to use negations, you may add a special constant formula "*Impossible"*. The task *Impossible* must have no solutions. It means that

$d : r$: Impossible : t

is always false. Then, the negation of B can be expressed as ($B \Rightarrow$ Impossible).

5 Ergatic Deductive Synthesis of Problems Solutions

By the synthesis of a solution of the problem B we mean the construction by some ergatic subject by the problem B of such solution r and the procedure for diagnostics d that for any admissible objects t,

$d{:}r{:}B{:}t$

It is assumed that by any formula B, Lr(B), Lt(B), Ld(B) are the languages of admissible solutions r, tests t and diagnostic procedures d, respectively.

A constructive approach involves the automatic extraction of r and d from some construction b, called the constructive proof of formula B or the deductive solution of problem B:

Xr(b,B)=r, Xd(b,B)=d

For functions Xr and Xd, there are also, generally speaking, Xrs and Xds executors and Xrp and Xdp programs respectively:

Xrs(Xrp,b,B)=r, Xds(Xdp,b,B)=d

To construct a deductive solution of problem B, the function Gen of constructing proofs is usually used. The function is applied to task B and produces some object that claims to be the deductive solution of the problem. The function Gen has also an executor Gens and a program Genp:

Gens(Genp,B)=b

Unfortunately, the procedure for constructing a deductive solution is often very complicated, requires the participation of people in the work of the executor and does not always end successfully.

In order to identify a failure, a special verification function *Check* is used:

Check(b,B)=1

if b is indeed a required deductive solution of problem B. Otherwise:

Check(b,B)=0

The function *Check* has an executor *Checks* (as a rule it is an automaton) and a program *Checkp*:

Checks(Checkp,b,B) = Check(b,B)

Here is an example of rules for checking deductive solutions (proofs) for problems written in the language of predicate logic without functions.

For convenience, we assume that the object to be proved (the problem being solved) has a *sequent* form (see, for example, [20]):

$$\boldsymbol{x}\ \boldsymbol{G} \Rightarrow B$$

where $\boldsymbol{x}$ is a string (array) of names, $\boldsymbol{G}$ is a chain (array) of formulas, B is a formula, $\boldsymbol{G}$ and B can use the names $\boldsymbol{x}$ as free variables. This sequent is equivalent to the formula:

$$\forall \boldsymbol{x}(\boldsymbol{G} \Rightarrow B)$$

We mean signs of conjunction between the formulas $\boldsymbol{G}$. If $\boldsymbol{x}$ and $\boldsymbol{G}$ are empty, then we simply solve the problem $\boldsymbol{B}$ without any parameters $\boldsymbol{x}$ and premises $\boldsymbol{G}$.

A deductive solution of the problem (the proof of the sequent)

$$\boldsymbol{x}\ \boldsymbol{G} \Rightarrow B$$

is an expression (term) t, the structure of which is described below. The statement

$$\text{Check}(t, \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)$$

we write for brevity, in the form

$$t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B$$

This term is an expression which, after interpreting in two different ways, can be transformed both into the actual solution of the problem and the definition of the diagnostic procedure.

We determine the structure of the deductive solution by induction from solutions of simple problems to solutions of more complex ones. We consider formulations of the simplest problems as axioms. The solution of the axiom in this system is given by the term pr(i), where i is a natural number:

$$(\text{pr}(i) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B) \Leftrightarrow G[i]{=}B$$

This means that among the premises of $\boldsymbol{G}$ on the i^{th} place there was the very conclusion (*pr* means "premise reproduction"). That is, you do not have to do anything, the task is solved in one step. One of the arguments is already the result of the solution. It remains only to enter this input to the exit.

Let us give examples of other rules for verifying the solution. Here is the rule for verifying the solution of a composite (conjunctive) problem (*cg* means "conjunction generation"):

$$(\text{cg}(t,u) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\&C)) \Leftrightarrow$$

$$(t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)\ \&\ (u : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow C)$$

It prescribes to solve the problem in parts.

Rules for verifying the solution of a disjunctive problem i.e. problem with the choice of a variant (*dg* means "disjunction generation"):

$$(\mathrm{dg0}(t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B \vee C)) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)$$

$$(\mathrm{dg1}(t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B \vee C)) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)$$

allow you to select the required option.

The rule of checking an implicative task, i.e. task with constructing a function (*ig* means "implication generation"):

$$(\mathrm{ig}(t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B \Rightarrow C)) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G}B \Rightarrow C)$$

extends the list of premises from $\boldsymbol{G}$ up to $\boldsymbol{G}B$ and changes the conclusion to C.

For problems with quantifiers, there are rules (*eg* – "existence generation", *ag* – "ALL generation"):

$$(\mathrm{eg}(i,t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow \exists y B(y)) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B(x[i])$$

$$(\mathrm{ag}(t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow \forall y B(y)) \Leftrightarrow (t : \boldsymbol{x}a\ \boldsymbol{G} \Rightarrow B(a))$$

where a is a new name not found among $\boldsymbol{x}$.

Next, we give the rules for verifying the use of premises.

The rule of checking the use of a complex structure, i.e. conjunction (*cu* – "conjunction usage"):

$$(\mathrm{cu}(i,t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow D) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G}BC \Rightarrow D)$$

where $G[i] = (B \& C)$, lengthens the list of premises, allowing the use of parts of the premise separately.

The rule for checking the use of the variant, i.e. disjunction (*du* – "disjunction usage"):

$$(\mathrm{du}(i,t,u) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow D) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G}B \Rightarrow D)\ \&\ (u : \boldsymbol{x}\boldsymbol{G}C \Rightarrow D)$$

where $G[i] = (B \vee C)$, leads to a verification of the analysis of two cases.

The rule for checking the use of the function, i.e. implication (*iu* – "implication usage"):

$$(\mathrm{iu}(i,t,u) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow D) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)\ \&\ (u : \boldsymbol{x}\ \boldsymbol{G}C \Rightarrow D)$$

where $G[i]=(B \Rightarrow C)$, considers the entire decision process divided into three stages: constructing the solution of B, applying the solution of $(B \Rightarrow C)$, and constructing the solution of D.

The rule of checking the use of the solution of the problem with the quantifier of existence (*eu* – "existence usage"):

$$(\mathrm{eu}(i,t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow D) \Leftrightarrow (t : \boldsymbol{x}a\ \boldsymbol{G}B(a) \Rightarrow D)$$

где $G[i]=\exists yB(y)$, a – новое имя, не встречающееся среди x.

The rule for verifying the use of a task with a quantifier of universality (*au* – "ALL usage"):

$$(\mathrm{au}(i,j,t) : \boldsymbol{x}\ \boldsymbol{G} \Rightarrow D) \Leftrightarrow (t : \boldsymbol{x}\ \boldsymbol{G}B(x[j]) \Rightarrow D)$$

where $G[i]=\forall yB(y)$, Adds a special case of the formula $\forall yB(y)$ to the premise.
Cases that are not covered by the above rules lead the procedure of verifying the solution to the negative result.

Let us demonstrate by examples how, in this case, the programs extracted from the deductive solution of the problem should work. At the input of the realization (task solution) of the sequent, two lists are given: a list of values of variables and a list of realizations of the premises. That is, if f is proposed as a solution of the problem

$$\boldsymbol{x}\ \boldsymbol{G} \Rightarrow B$$

then $f(\boldsymbol{x},\boldsymbol{z})$ must be a solution of the problem B, where $\boldsymbol{z}$ –are realizations of $\boldsymbol{G}$. The test for this sequent consists of the following parts: the values of the variables $\boldsymbol{x}$, diagnostics $\boldsymbol{y}$, realizations $\boldsymbol{z}$ of premises $\boldsymbol{G}$, and test b for the task B. The diagnostics of the sequent under consideration consists of two functions: g и h. In this case, the value of $g(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\mathrm{b})$ is proposed as a test list for premises $\boldsymbol{G}$, and the value of $h(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ is proposed as diagnostics of B.

We assume the following formula for verifying the solution of the sequent

$$(g,h) : f :\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B :\ (\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},\mathrm{b})$$

if from the fact that for any i

$$y[i]{:}z[i]{:}G[i]{:}g(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)$$

it follows that $h(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}){:}f(\boldsymbol{x},\boldsymbol{z}){:}B{:}b$

It is possible for convenience to break the diagnostic extraction procedure Xd into two parts:

$$(g,h)=\mathrm{Xd}(p,\boldsymbol{x}\ \boldsymbol{G}\Rightarrow B)$$

$$g=\mathrm{Xdg}(p,\boldsymbol{x}\ \boldsymbol{G}\Rightarrow B),\ h=\mathrm{Xdh}(p,\boldsymbol{x}\ \boldsymbol{G}\Rightarrow B)$$

The explanation of how the extraction procedures Xr and Xd should work, we start with the axioms:

$$\mathrm{Xr}(\mathrm{pr}(i),\ \boldsymbol{x}\ \boldsymbol{G}\Rightarrow B)(x,z)=z[i]$$

$$\mathrm{Xdg}(\mathrm{pr}(i)\ ,\boldsymbol{x}\ \boldsymbol{G}\Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)[i]=y[i]$$

$$\mathrm{Xdh}(\mathrm{pr}(i),\boldsymbol{x}\ \boldsymbol{G}\Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)=b$$

In the second line, the components of the result other than the i-th one are not important and may be chosen arbitrary.

We also show the procedures for extracting solutions and their support for some more complex cases. Here are the formulas for the operator *cg* (operator of generating conjunction, that is, a composite problem):

$\mathrm{Xr}(\mathrm{cg}(t,u)\ ,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\&C))(\boldsymbol{x},\boldsymbol{z}) =$

$\quad (\mathrm{Xr}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{z}),\ \mathrm{Xr}(u,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow C)(\boldsymbol{x},\boldsymbol{z}))$

$\mathrm{Xdg}(\mathrm{cg}(t,u),\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\&C))(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},(b,c))[i] =$

$\text{if } \mathrm{Arb}(\boldsymbol{y},\boldsymbol{z},G[i],\mathrm{Xdg}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)[i])$

$\text{then } \mathrm{Xdg}(u,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow C)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},c)[i]$

$\text{else } \mathrm{Xdg}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)[i]$

$\mathrm{Xdh}(\mathrm{cg}(t,u),\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\&C))(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) =$

$\quad (\mathrm{Xdh}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}),\ \mathrm{Xdh}(u,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow C)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))$

As you can see, the diagnostic procedure actively uses the verification procedure Arb. Here are the formulas for the operator dg0:

$\mathrm{Xr}(\mathrm{dg0}(t),\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\vee C))(\boldsymbol{x},\boldsymbol{z}) = (0,\mathrm{Xr}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{z}))$

$\mathrm{Xdg}(\mathrm{dg0}(t),\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\vee C))(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},(b,c))[i] =$

$\quad \mathrm{Xdg}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z},b)[i]$

$\mathrm{Xdh}(\mathrm{dg0}(t),\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow (B\vee C))(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}) =$

$\quad (\mathrm{Xdh}(t,\ \boldsymbol{x}\ \boldsymbol{G} \Rightarrow B)(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}),\mathrm{dummy}(C))$

where dummy(C) is an arbitrary function suitable for diagnostics of C by type (the result of its operation is not important).

Conclusions

The aim of the work was to obtain novel knowledge and technologies, in the creation of IT support, for descriptive and constructive activities. The key focus is directed to the theoretical and technological knowledge related to the manifestation of symbiosis, as well as, to various computing and intellectualization of activities.

Methods for constructing algorithms and computer programs that take into account aspects of counteraction, detection and elimination of errors are investigated.

The transition from classical programming, as a planning of a clear order of actions to the development of task formulations, in particular, in the framework of constructive logic based on the classification of complexity and management of complexity, resource intensity, as well as, the transition to ergatic computing, is explored.

A variant of constructive understanding is also studied: The practical understanding in the face of opposition and support.

References

[1] Volfengagen V. E.: Applicative computing: attempts to establish the nature of computations (Applikativnyy komp'yuting: popytki ustanovit' prirodu vychisleniy). In: V. I. Konov (ed) Sbornik nauchno-populyarnykh statey — pobediteley konkursa RFFI 2006. V. 10. Oktopus, Priroda, Moscow. 2007, pp. 446-459

[2] Rudenskiy O. V., Rybak O. P.: Innovative civilization of the XXI century: convergence and synergy of NBIC-technologies. Trends and forecasts 2015-2030 (Innovatsionnaya tsivilizatsiya XXI veka: konvergentsiya i sinergiya NBIC-tekhnologiy. Tendentsii i prognozy 2015-2030). Informatsionno-analiticheskiy byulleten' 2010. 3

[3] Beltiukov A. P.: Intuitionistic formal theories with realizability in subrecursive classes. *Annals of Pure and Applied Logic*, 1997, 89: pp. 3-15

[4] Beltiukov A. P.: A strong induction scheme that leads to polynomially computable realizations. *Theoretical Computer Science*, 2004, 322: pp. 17-39

[5] Beltiukov A. P.: Small complexity classes and automatic deductive synthesis of algorithms (Malyye slozhnostnyye klassy i avtomaticheskiy deduktivnyy sintez algoritmov). *Izvestiya instituta matematiki i informatiki UdGU*, Izhevsk, Russia, 1995, 2: pp. 3-89

[6] Maslov S. G., Beltu'kov A. P.: Active layered terminology system" (Aktivnaya rassloyennaya terminologicheskaya Sistema). *Ustoychivoye innovatsionnoye razvitiye: proyektirovaniye i upravleniye*. Dubna, 2011, Vol. 7, 3(12): pp.103-113

[7] Kovacs G.: Role of artificial intelligence and robots in digital production and beyond. In: *Proceedings of the 18th international workshop on computer science and information technologies* (*CSIT'2016*), Vol. 1, USATU, Ufa, Russia, 2016, pp. 40-44

[8] Maslov S. G., Beltu'kov A. P., Morozov O. A.: IT-sphere Constructive Directions. In: *Proceedings of the 11th international workshop on computer science and information technologies (CSIT-2009)*, UGATU, Ufa, Russia, 2009, pp. 131-135

[9] Beltu'kov A. P., Maslov S. G.: On the problems of readers and the process of reading in an electronic environment (O problemakh chitateley i protsesse chteniya v elektronnoy srede). *Vestnik Udmurtskogo universiteta*. *Mathematics. Mechanics. Computer science*. UdGU, Izhevsk, Russia, 2010, 4: pp.101-111

[10] Kosikov S. V.: Information systems: a categorical approach (Informatsionnyye sistemy: kategornyy podkhod). YurInfoR-Press, Moscow, 2005

[11] Kovalev S. P.: The theory-categorical approach to the design of software systems (Teoretiko-kategornyy podkhod k proyektirovaniyu programmnykh system) *Fundament. i prikl. matem.* 2014, 19:3: pp. 111-170

[12] Antonov V. V.: The method of designing an adaptive software package based on the methodology of the formal model of an open domain (Metod proyektirovaniya adaptivnogo programmnogo kompleksa na osnove metodologii kategoriynoy formal'noy modeli otkrytoy predmetnoy oblasti) *Vestnik UGATU*, 2015, Vol. 19, 1(67): pp. 258-263

[13] Sergeev S. F.: Autopoietic basis of biotechnosphere evolution (Autopoeticheskiy bazis evolyutsii biotekhnosfery). *Obrazovatel'nyye resursy i tekhnologii*, 2015, 1(9): pp. 57-65

[14] Mikheev M. Yu., Zhashkova T. V., Semochkina I. Yu., Roganov V. R, Shcherban A. B.: Mathematical and information-structural models of ergatic systems (Matematicheskiye i informatsionno-strukturnyye modeli ergaticheskikh sistem). Penza, 2015

[15] Sergeev S. F.: Humans within the Framework of Technobiotic Evolution *Russian Federation European Journal of Psychological Studies*, 2014, Vol. 3, 3 pp. 93-101

[16] Levenchuk A. I.: System-Engineering Thinking (Sistemnoinzhenernoye myshleniye). *Electronic resource*, (30.05.2015), http://techinvestlab.ru/systems_engineering_thinking

[17] Chudova N. V.: Understanding: the subject of research and the object of modeling (Ponimaniye: predmet issledovaniya i ob"yekt modelirovaniya) *Artificial Intelligence and Decision Making (Iskusstvennyy intellekt i prinyatiye resheniy)*, 2012; 4: pp. 3-31

[18] Markov A. A.: An attempt to construct the logic of constructive mathematics (Popytka postroyeniya logiki konstruktivnoy matematiki). In: *Studies in the theory of algorithms and mathematical logic (Issledovaniya po teorii algorifmov i matematicheskoy logike)*, Moscow, 1976, Vol. 2, pp. 3-31

[19] Shanin N. A.: On the hierarchy of ways of understanding statements in constructive mathematics. *Tr. Matem. in-ta AN SSSR*, 1973, т. 129, pp. 203-266

[20] Kleene S. C.: Introduction to metamathematics, North-Holland, 1951

[21] Gödel, K.: Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. In: Feferman, S., Dawson, Jr., J. W., Kleene, S. C., Moore, G. H., Solovay, R. M., and van Haijenoort, J. (ed) *Collected Works*. Oxford University Press, 1990, vol. II, pp. 240-251

[22] Spector C.: Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics. Recursive Function Theory. *Proc. Symposia in Pure Mathematics.* 1962, pp. 1-27

[23] de Paiva V.: The Dialectica Categories. University of Cambridge, Computer Laboratory, PhD Thesis, Technical Report 213, 1991

[24] Avigad J, Feferman S.: Gödel's Functional ("Dialectica") Interpretation. In: Buss S. R. (ed) *Handbook of Proof Theory*, Elsevier Science, 1999

[25] Shirahata M.: The Dialectica Interpretation of First-order Classical Affine Logic. *Theory and Applications of Categories*, 2006. Vol. 17, 4: pp. 49-79

[26] Troelstra, A. S., Smoryński C. A., Zucker J. I., Howard W. A.: Metamathematical Investigation of intuitionistic Arithmetic and Analysis. Springer Verlag, Berlin. 1973. pp. 1-323

[27] "Big Data Meets High Performance Computing" (WHITE PAPER Intel Enterprise Edition for Lustre* Software High Performance Data Division), 2014. https://www.intel.com/content/www/us/en/lustre/intel-lustre-big-data-meets-high-performance-computing.html

[28] Fox G.: High Performance Computing and Big Data, 2016. - https://www.slideshare.net/Foxsden/high-performance-computing-and-big-data

[29] "NIST Big Data Interoperability Framework". 2015. - http://dx.doi.org/10.6028/NIST.SP.1500-3

[30] Basin D., Deville Y., Flener P. Hamfelt, A., Nilsson J. F.: Synthesis of Programs in Computational Logic. In: Bruynooghe M., Lau K.-K. (Eds.) *Program Development in CL*, 2014, LNCS 3049: pp. 30–65

[31] Maslov S. G., Beltiukov A. P.: Problems of constructing a natural scientific network of knowledge of descriptive and constructive activity (Problemy postroyeniya yestestvenno-nauchnoy seti znaniy deskriptivno-konstruktivnoy deyatel'nosti). In: *XII Vserossiyskoye soveshchaniye po problemam upravleniya. VSPU-2014*, IPU RAN, Moscow, Russia, 2014, pp. 639-640

[32] Maslov S. G., Beltiukov A. P.: Ergatic network as the basis of professional activity in the paradigm of sustainable development (Ergaticheskaya set' kak osnova professional'noy deyatel'nosti v paradigme ustoychivogo razvitiya). In: Electronic scientific publication "*Sustainable innovation development: design and management*". Dubna, 2015. Vol. 11. No. 1(26): pp. 36-50. http://www.rypravlenie.ru/wp-content/uploads/2015/04/03-Maslov_Beltyukov.pdf

[33] Beltiukov A. P., Maslov, S. G.: Organizing understanding in the framework of ergatic system In: *Proc. of 17th International Workshop on Computer Science and Information Technologies (CSIT'2015)*, Vol.1, USATU, Ufa, Russia, 2015, pp. 60-64

[34] Maslov S. G., Beltiukov A. P.: On the problems of intellectual activity in ERGANET (O problemakh intellektual'noy deyatel'nosti v ERGANETE) In: *Materials of the 8th All-Russian Multiconference (Materialy 8-y Vserossiyskoy mul'tikonferentsii)*. 3 Vol. Rostov-on-Don: Southern Federal University Publishing House, 2015, pp. 142-144

[35] Beltiukov A. P., Maslov, S. G.: Logical-deductive synthesis of program-information objects in decision-making systems (Logiko-deduktivnyy sintez programmno-informatsionnykh ob''yektov v sistemakh prinyatiya resheniy). In: *ITIDS'2016: Proceedings of the 4th International Conference on Information Technologies for Intelligent Decision Making Support.* Vol. 1, USATU, Ufa, Russia, 2016, pp. 25-30

[36] Maslov S. G., Beltiukov A. P.: Current situation and trends in the design of IT systems (Sovremennaya situatsiya i tendentsii v proyektirovanii IT-sistem). In: *ITIDS'2016: Proceedings of the 4th International Conference on Information Technologies for Intelligent Decision Making Support*. Vol. 1, USATU, Ufa, Russia, 2016, pp. 150-154

[37] Maslov S. G., Beltiukov A. P.: Information and technological support of ergatic intelligence. CSIT'2016: Proceedings of the 18th International Workshop on Computer Science and Information Technologies, Czech Republic, Prague-Kunovice, September 26-30, 2016, V.1, – Ufa: Ufa State Aviation Technical University, 2016, pp. 40-44

[38] Beltiukov A. P., Maslov, S. G.: On understanding within the framework of the ergatic system (O ponimanii v ramkakh ergaticheskoy sistemy). In: *International cooperation: Integration of educational spaces. Materials of the III International Scientific and Practical Conference (Mezhdunarodnoye sotrudnichestvo: Integratsiya obrazovatel'nykh prostranstv)*. UdSU, Izhevsk, 2016, pp. 145-151

[39] Alpuente M., Cuenca-Ortega A., Escobar S., Meseguer J.: Partial Evaluation of Order-Sorted Equational Programs Modulo Axioms. In: Hermenegildo M., Lopez-Garcia P. (eds) Logic-Based Program Synthesis

and Transformation. LOPSTR 2016. Lecture Notes in Computer Science, vol 10184 (2017) Springer, Cham, pp. 3-20

[40] Albert E., Bezirgiannis N., de Boer F., Martin-Martin E.: A Formal, Resource Consumption-Preserving Translation of Actors to Haskell. In: Hermenegildo M., Lopez-Garcia P. (eds) Logic-Based Program Synthesis and Transformation. LOPSTR 2016. Lecture Notes in Computer Science, vol 10184 (2017). Springer, Cham, pp. 21-37

[41] De Angelis E., Fioravanti F., Meo M.C., Pettorossi A., Proietti M.: Verification of Time-Aware Business Processes Using Constrained Horn Clauses. In: Hermenegildo M., Lopez-Garcia P. (eds) Logic-Based Program Synthesis and Transformation. LOPSTR 2016. Lecture Notes in Computer Science, vol 10184 (2017). Springer, Cham, pp. 38-55

[42] Alpuente M., Pardo D., Villanueva A.: Symbolic Abstract Contract Synthesis in a Rewriting Framework. In: Hermenegildo M., Lopez-Garcia P. (eds) Logic-Based Program Synthesis and Transformation. LOPSTR 2016. Lecture Notes in Computer Science, vol 10184 (2017). Springer, Cham, pp. 187-202

[43] Tarau P.: A Hiking Trip Through the Orders of Magnitude: Deriving Efficient Generators for Closed Simply-Typed Lambda Terms and Normal Forms. In: Hermenegildo M., Lopez-Garcia P. (eds) Logic-Based Program Synthesis and Transformation. LOPSTR 2016. Lecture Notes in Computer Science, vol 10184 (2017). Springer, Cham, pp. 240-255