The Theory of S-symbols in the Methodological Arsenal of Artificial Intelligence

Vladimir D. Ilyin

***Abstract* — The theory of S-symbols is an extended generalization of the theory of S-modeling. It is considered as a part of the methodological support for the development of artificial intelligence systems in the S-environment (including knowledge systems, systems of S-modeling of problems and program construction, etc.). The S-environment, based on interconnected systems S-(symbols, codes, signals), serves as the infrastructural basis for the implementation of information technologies for various purposes. The rationale for expediency and basic notions (S-symbol, S-code, S-signal, etc.) is provided. The kinds and types of S-(symbols, codes and signals) are defined. Equivalence, order, and membership relations are introduced and defined on systems of S-(symbols, codes, and signals). Definitions of notions related to S-problem objects (S-problem, P-graph, etc.) are provided. The basics of designing S-problem objects (including the construction of resolving structures on P-graphs) are described. The classes of basic S-problems are defined. Definitions are accompanied by examples.**

***Keywords* — S-symbol, S-code, S-signal, S-environment, S-problem, P-graph, Resolving Structure, Programming Automation.**

I. Introduction

Without effective symbolic modeling [1], [3]–[5], [21] of the studied entities, the development of science, technology and other types of intellectual activity is impossible. An example of an effective symbolic representation of *hypertext network*[2] *resources*[3] is the Web [2]. When studying the world and themselves, people build *symbolic* (*audio*, *video*, etc.) *models* of entities that reflect the objects being studied and the relationships between them [3]–[5]. These models are represented by interaction, specification and programming languages; *digital twins* [7]–[12] of various objects, etc. Symbolic models serve not only as a means of cognition, but also as tools that help invent artificial things, expanding and improving the natural capabilities of a person.

Achieving a goal in the course of some activity implies a clear idea of the problems to be solved. When achieving non-trivial goals (such as designing a machine, developing information technology, etc.), the symbolic representation of the idea in the *human-machine environment for problem solving* (*S-environment*) [3]–[5] is the most productive. This approach has a number of proven advantages. Firstly, by analyzing the S-model of the idea (device scheme, specification of problem, etc.), we can check whether the S-model corresponds to the idea, and if it does not, make corrections to the symbolic model. Secondly, if the symbolic model is recognized as corresponding to the idea, it is possible to verify the validity of the idea itself. If verification is successful, a decision on the feasibility of implementing the idea can be made. Otherwise, engaging in a change of idea is necessary. The idea of achieving a goal using S-models is embodied in many technologies for various types of activities. The most successful technologies prohibit behavior that does not comply with the rules (attempts of unacceptable behavior have as much chance of success as attempts to play poker against a chess program).

The *theory of S-symbols* [3]–[4] is extended generalization of the *theory of symbolic modeling* (*S-modeling*) [5]. It includes conceptual fundamentals with definitions of the basic notions [*S-*(*symbol*, *code*, *signal*), *S-problem*, etc.], methodology for *formalizing knowledge about S-problems*, a model of a *network S-problem solver* and definitions of *classes of basic S-problems* [*S-*(*representations*, *transformations*, *recognition*, *interpretation*, *construction*, *exchange*, *preservation*, *accumulation*, *search*, *information protection* and *cryptography*)]. *The basic S-problems* are called, to the solution of which other S-problems are reduced. The key S-problems of each class are considered.

Nowadays, people and computer devices operate in a *symbol-code-signal S-environment* [3]–[5], which in modern implementation is digital. Machine-to-Machine technologies (M2M) [13]–[14]; cloud computing technologies and electronic services [15]; the Internet of Things [16] and digital twins are successfully used to solve problems in various fields of activity.

A. Markuping Text Fragments and Writing Formulas

To record formulas and highlight definitions, comments and examples as part of S-messages, *TSM-complex language tools* (*TSM*: *textual S-symbolic modeling*) are used [3]–[5]. *TSM* has the means of writing formalized expressions (without using formula editors), highlighting parts of S-messages and replacing frequently repeated fragments with selected abbreviations. To fulfill the descriptions, a standard keyboard and a set of tools available in the text editors of the LibreOffice, OpenOffice, or other packages are sufficient.

The article uses the following TSM-means of highlighting text fragments:

- □ <description fragment> □ ≈ definition (hereafter the symbol ≈ replaces the word "means");
- ◊ <description fragment> ◊ ≈ remark;
- ∘ <description fragment> ∘ ≈ example.

[1] Manuscript received Nov 14, 2023.

Vladimir D. Ilyin, Federal Research Center "Computer Science and Control" of Russian Academy of Sciences, 44/2, Vavilova Street, 119333, Moscow, Russia (e-mail: vdilyin@yandex.ru)

[2]Ilyin V.D. Computer network. The Great Russian Encyclopedia https://bigenc.ru/c/komp-iuternaia-set-e016ec

[3]Information resources. Encyclopedia Runiversalis https://руни.рф/Информационные_ресурсы

For set-theoretic formulas, the following form of notation is used:

- $A * B \approx$ Cartesian product of sets A and B;
- $R < A * B \approx$ binary relation defined on sets A and B.

Italics and bold are used to highlight the first occurrences of the names of notions and fragments of the description to which the author wants to draw attention.

II. BASIC NOTIONS

□ *S-objects* encompass *S-*(*symbols*, *codes*, *signals*) and *S-symbol constructions* (∘ specification languages, programs, etc. ∘). □

A. S-symbols

□ An *S-symbol* is regarded as a substitute for a natural or invented object, designating this object and serving as an element of a certain system for constructing *S-messages* (∘ video, audio messages, etc.; the text of an electronic document, an electronic geographical map, a video clip – all these are S-messages. ∘) in an S-environment designed for perception by a person or an *S-machine* [computer or computer device (smartphone, digital camera, etc.)]. □

∘ Braille for the blind is a system of text S-symbols for constructing text messages designed for perception by touch with fingers; musical notation, a system of musical S-symbols is a means of constructing musical audio messages presented in graphic form. The system of S–symbols of chess notation is a means of recording chess games in the form of text messages. The alphabet, together with punctuation marks, forms a system of text S–symbols for constructing messages according to the rules of the grammar of the language (and each element of the alphabet is a substitute for the sound used in speech messages). ∘

The kinds of S-symbols are defined, each of which has *types of S-symbols*.

□ *The kind of S-symbols* corresponds to the means of receiving messages by a person (or robot): *visual* – eyes; *audio* – ears; *tactile* – fingers; *smell* – nose. □ To produce S-messages, a person can use speech organs, body parts that produce distinguishable movements (hand gestures, finger movements, etc.), and eyes (selecting an object, indicating the place of its new placement, etc.).

◊ In the modern (early 2020s) S-environment, vision (perception of text, fixed and moving images, etc.) and hearing (perception of speech, music, etc.) are used relatively productively. Touch is utilised in mobile phones (for receiving vibration calls), in gaming devices, etc. Smell is in the stage of experimentation (devices for recognizing odors are produced). ◊

□ *The type of S-symbols* corresponds to a set of symbols for which a set of attributes and a family of valid operations are defined. □

The S-visual kind corresponds to the types: *S-graphic* [for constructing messages containing still images (photographs, diagrams, etc.)]; *S-video* (for constructing messages containing moving images), etc. ∘The Periodic Table (on the web page) is an S-graphical representation of the S-message about the periodic law (discovered by Mendeleev D. I.), which established the dependence of the properties of chemical elements on their atomic mass. ∘

The types correspond to *the S-audio kind*: S-speech (for constructing S-messages containing speech fragments); S-music, etc.

The S-tactile kind corresponds to the types: *S-kinetic* [is used to generate S-messages by moving elements of devices designed to interface with the S-machine (∘mobile phone vibration – an S-message about an incoming call ∘)]; *textured* [for implementing S-messages by changing the surface texture of device elements (∘flat → wavy or ribbed, etc. ∘)]; *S-thermal* (for implementing S-messages by changing the surface temperature of interface device elements).

Elementary and Composite S-objects. □ An S-object used to construct *composite S-objects* is called an *elementary S-object*. □ ∘A pixel is an elementary graphic S-object of raster graphics devices. A pictogram is a composite graphic S-object. A hyperlink is a composite graphic S-object derived from two S-objects [∘a text (or image) fragment of one S-object and the URL of a text (or image) fragment of another S-object]. ∘

B. S-signals

□ An *S-signal* is a physically realized representation of an S-symbol (or an S-symbolic message), or an S-code (or an S-code message), designed for S-(sending, reception, recognition, interpretation, construction, copying and search) by a person or S-machine. ∘In the S-environment, a person (or robot) receives S-messages implemented in the form of optical, sound, etc., S-signals. Technical devices of S-machines are designed to work with optical (∘digital photo and video cameras ∘), electrical (∘ microprocessors of S-machines ∘), etc. ∘

C. S-codes

□ An *S-code* serves as a substitute for an S-object [∘S-(symbol, symbol message, signal, signal message, code or code message). It is used to represent them in S-machines. An S-code is intended for construction, transformation, storing, interpreting, exchange of S-messages and manipulating them in the S-environment. □

∘ Graphic S-(symbols and codes) of the planets of the Solar system are used by astronomers in electronic documents [Planet Mercury corresponds to the S-symbol ☿ and the S-code – U3+263F (Unicode Standard)]. Morse code is an example of sound coding for texts. Text encoding using Morse code is employed to send text messages. For the sending of Morse-coded messages, the long-standing technology of the auditory radiotelegraphy is most common. In the Navy, Morse-coded messages are also sent using light signals. Nowadays, some smartwatch models use tactile signals to send Morse-coded messages.

QR code[4] is a relatively modern example of graphic text encoding, which has many applications (ranging from payment technologies to technologies of object positioning in augmented reality. ∘

[4]What are QR codes and how to scan them https://www.kaspersky.ru/resource-center/definitions/what-is-a-qr-code-how-to-scan

◊ *Digital encoding of S-messages* allows the application of methods for solving problems that can be represented in the form of programs designed to be executed by digital S-machines. When digitally encoding S-objects, they are put in one-to-one correspondence with numbers that can be efficiently represented in S-machine memory. Digital encoding of S-signals and S-signal messages [∘using analog-to-digital converters (ADCs) ∘] is at the heart of the construction of modern communication systems, data processing, etc. One exampl of effective digital encoding of S-messages is the *CDMA*[5] family of mobile communication technologies (providing simultaneous operation of many channels in a common frequency band by encoding radio signals). ◊

D. S-objects Relations

□ *S-equivalence* defines the interchangeability of S-objects. The S-equivalence relation can be set for any number of S-objects (at least two). □ ∘A mobile phone vibration call (an S-message composed of tactile S-symbols) is equivalent to any of the audio calls (∘messages composed of musical S-symbols). ∘

□ *S-order* defines the sequence of S-objecs. □ ∘S-order is given on the set of elements of the alphabet. ∘

□ *S-membership* defines the occurrence of an S-object in some set of S-objecs. □ ∘The S-symbol □ used in the theory of S-symbols to markup S-text fragments. It belongs to the set of S-symbols of the TSM-complex. ∘

E. S-objects Typing

□ *S-type X* is a set of *X* S–objects whose elements have a fixed set of attributes and a family of valid operations. It can have *S-subtypes* called *S-type X specialization*s and *S-supertypes* called *S-type X generalizations*. □

□ *Specialization of the S-type X* is the generation of *the S-subtype X* [::*rule*] (here the double colon "::" is the symbol of specialization) with a family of relations, expanded by the addition of the *rule* relation (selects a subset *X* [::*rule*] of the set *X*). Specialization is also called the result *X* [::*rule*] of this generation (*X* [::*rule*] < *X*). □∘Type *S-text* is a *specialization of type S-graphic*. ∘ S -type specialization, given by a sequence of added relations *X* [::(*rule* 1)::*rule* 2], is called *a specialization of type X* [::*rule* 1] by the relation *rule* 2. The number of specializing relations in the sequence is unlimited. In this case, the names of the relations preceding the last one are enclosed in parentheses, and before the opening bracket of each pair of brackets is a double colon.

□ *Generalization of S-type Z* is the generation of its *S-supertype Z* [#*rule*] by weakening (here # is the weakening symbol) the *rule* relation from the family of relations corresponding to type *Z*. Exclusion of *rule* is its complete weakening. □

[5]CDMA (Cellular Mobile Communication System) https://www.ixbt.com/mobile/sys-cdma.html

F. S-system of Notions

□ The definition of *the S-system of notions* is a description of its S–model, accompanied by *the specification of applicability*. ‹ *S-system notions Sc* › ≈ ‹ S^{Sc} ≈ collection of notions on which the S-object being studied ›, ‹ *rel* (S^{Sc}) ≈ the family of relations defined on S^{Sc} ›. □

The definition is addressed to researchers and developers of information technologies. It is represented in the form of an S-message, designed for exchange, storing, accumulation and search in the S-environment. For each notion of system *Sc*, a set of values is defined.

The *specification of applicability* of the definition is given by the description of the types:

- *the correspondent* (for whom the definition may be useful);
- *the purpose*, in the process of achieving the one the definition is advisable to be applied (∘ problems, in the study of which the definition may be useful ∘);
- *the stage* where it makes sense to use the definition (∘ problem statement, development of a solution method ∘). □

∘ System *tr* of notions, named *a triangle,* ≈ ‹ set^{tr} ≈ collection of notions ›, ‹ *rel* (set^{tr}) ≈ family of relations defined on set^{tr} ›. In *tr*, the elements of set^{tr} are the sides (*a*, *b*, *c*), angles (*α*, *β*, *γ*), perimeter *p*, etc. The family *rel* (set^{tr}) includes $p = a + b + c$; $\alpha + \beta + \gamma = \pi$, etc.∘ ∘ The system $tr^{\pi/2}$ of notions of a right triangle can be defined as a *specialization of tr*: $tr^{\pi/2} \approx tr$ [:: $\alpha = \pi/2$] (by adding the restriction $\alpha = \pi/2$, which distinguishes a subset of triangles where the value of one of the angles is equal to $\pi/2$). ∘

G. S-systems of Knowledge

□ *The S-model of the knowledge system Sk* ≈ ‹ *Ca* ≈ S-model of the notions system *Sc* ›, ‹ set^{lng} ≈ the set of message languages interpreted on *Ca* ›, ‹ set^{intr} ≈ the set of interpreters of *Ca*-messages composed in languages from set^{lng} ›.

Interpreting the S-message on *Ca*:

1. Constructing an output message based on a given input message (messages are represented in languages from set^{lng});
2. Analysis of the output message (whether changes are required in *Ca*);
3. If required, initiate changing the *Ca*; if not, end. □

H. S-(Message, Data, Information)

□ *S-message* is a finite sequence of S-(symbols, codes or signals) designed for recognition and interpretation by the recipient, that meets the requirements for *basic S-problems* solvers (see section 4). □

∘ S-systems of notions and knowledge which present the results of the study of certain entities (objects of research); computer programs; web pages and document files – all these are S-messages. ∘

□ *S-file* is a named unit of storage of the S-machine message code (data or program) on a drive (SSD, hard disk, etc.) of computer device (desktop, laptop, smartphone, digital camera, etc.). □

□ *S-data* is S-message required to solve a certain problem or a family of tasks, presented in a form designed for recognition, transformation and interpretation by S-problem solver (human or robot). □

□ *S-information* is the result of interpreting a S-message on the S-system of notions.

To extract information from a S-message, it is necessary to have:

- an accepted S-message presented in a form designed for recognition and interpretation by the S-message recipient;
- S-systems of notions stored in memory of recipient, among which is the necessary one for interpretation of the received S-message;
- mechanisms for finding the necessary S-system of notions, interpreting the S-message, presenting the result of interpretation in the form of a S-message and writing it to memory. □

∘ The result of interpretation of the m^a message presented in language *a*, received by the translator (human or robot) – translated to the m^b message in language *b*, is the information extracted from the m^a message. ∘

III. KNOWLEDGE ABOUT S-PROBLEMS

Representation of relations between notions in the form of solvable S-problems [*S-*(*representations*, *recognition*, etc.)] is a necessary condition for constructing methodologically significant problem *S-objects* (*composite S-problems*, *S-problem graphs*) in the S-environment [3]–[5].

The author's approach to modeling problems (from an arbitrary subject area) at the initial stage (1986-89) of research was formed as an alternative to the approach implemented in *the instrumental programming system Prize* [17]. This system traces the desire of developers to follow the methodology of automatic program synthesis popular in the 1980s [18]–[20]. Adherents of automatic program synthesis consider the specifications of problems and the process of designing programs (by formal transformations of the problem specification) as mathematical entities.

An experienced software system developer (who has experience in developing successfully operated software systems) knows that the specification of a non-trivial problem and the program relate in the same way as design documentation (in engineering) and a machine for the production of certain products. The suitability of the manufactured machine (its compliance with the design documentation) can be determined only with the help of tests developed by experts who know well what the subject should be able to do and what properties (including reliability and safety) should have [21]–[26].

A. S-problem Objects

□ *S-problem* ≈ {*Formul*, *Rulsys*, *Alg*, *Prog*}, where *Formul* is the statement of the S–problem; *Rulsys* is a set of systems of *mandatory and orienting requirements for solving the S-problem* [26] set in accordance with *Formul*; *Alg* is the union of sets of algorithms, each of which corresponds to one element from *Rulsys*; *Prog* is the union of sets of S-programs, each of which corresponds to one of the elements of *Alg* [4]–[5].

Formulation of the S-problem *Formul* ≈ {*Mem*, *Rel*}, where *Mem* is the set of notions of the problem on which the partition of *Mem* is given [*Mem* = *Inp* ∪ *Out* (*Inp* ∩ *Out* = ø)] and the set of relations *Rel* between notions defining the binary relation *Rel* < *Inp* * *Out*.

The set *Mem* is called *the S-problem memory*, and *Inp* and *Out* are its input and output, the values of which are supposed to be given and searched, respectively.

In general, the *Rulsys*, *Alg* and *Prog* may be empty: the numbers of their elements depend on the degree of knowledge of the S-problem.

For each element from *Rulsys*, *Alg* and *Prog*, *the specification of applicability* is set.

Descriptions of *Rulsys* elements include the specifications of *the types of S-problem solver* (∘ autonomous S-machine, network cooperation of S-machines, cooperation ‹ man – S-machine>, etc. ∘), *information security requirements*, etc.

Descriptions of *Alg* elements include *data on the requirements for the result obtained* (◊ for mathematical S-problems, among *the mandatory requirements* for the result may be a *proof of its existence* ◊), on *the permissible operating modes of the S-problem solver* (∘automatic local, automatic distributed, interactive local, etc. ∘), etc.

Descriptions of the use of S-programs include *data on programming languages*, *operating systems*, etc. ◊ Each S-program is accompanied by hyperlinks to *collections of test cases* and *examples of its application*. ◊

The S-algorithm is a system of rules for solving an S-problem (corresponding to one of the *Rulsys* elements), which allows for a finite number of steps to put the result set belonging to *Out* in unambiguous correspondence to a given data set belonging to *Inp*.

S-program is an S-message defining the behavior of an S-machine solver of an S-problem with specified properties. S-algorithm implemented (in a high-level programming language and/or a machine-oriented language and/or a machine command system). S-program in S-(symbolic, code) represents connected by translation relations. □

□ *S-data* is an S–message required to solve some S-problem or a set of S-problems, presented in a form designed to be received, recognized, transformed and interpreted by an S-problem solver (an S-machine program or a person). *S-*(*specialization of the message*) (*S-message*) by the parameter *recipient of the S-message* (*S-recipient*), the value of which is the *solver of the S-problem* (*S-solver*): *S-data* ≈ *S-message* [::*S-recipient* = *S-solver*] [3]–[5]. □

B. S-problem Objects Constructing

Memory connections between S-problems are determined by three types of functions, each of which is a function of two arguments and allows you to match a pair of S-problems with some third S-problem constructed from this pair.

S-problem a is connected to S-problem b by memory if there is at least one pair of elements {*elem* Mem^a, *elem* Mem^b} belonging to the memory of Mem^a of S-problem *a* and memory of Mem^b of S-problem *b*. At the same time, the same sets of values are given for the connected elements (elem Mem^a and *elem* Mem^b).

If *J* and *H* are sets of S–problems, $D \le J * J$ and each pair $\{j^i, j^k\}$ of elements from *D* corresponds to a certain element

from *H*, then *a memory connection function* $h \approx conn\ (j^i, j^k)$ is given. In this case, we call *D the domain of the definition of the function conn* and denote D^{con}. The set $R \approx \{h$: *elem H*; $h \approx conn\ (j^i, j^k)$; j^i: *elem* D^{con}, j^k: *elem* $D^{con}\}$ is called *the domain of conn function values*.

The type of connection from memory of S-problems depends on the content of the intersection from memory:

- whether the connection is made up of the elements of the output of one and the input of another S-problem;
- or from the elements of the outputs of S-problems;or from the elements of their inputs;
- or the connection is obtained by a combination of the previous methods.

□ *An elementary S-problem construction* is an S–problem pair. Any S-problem construction, in turn, can be used as a component of an even more complex S-problem construction. □

□ *System pS of knowledge about S-problem objects* (hereinafter referred to as *P-objects*) ≈ {*pA*, *Lng*, *Intr*}, where *pA* is the S-problem domain, *Lng* is the specification language for P-objects, *Intr* is an interpreter for the specifications of the desired P-objects in *pA*. □

If *P* is a set of P–objects, and *A* < *P* is its nonempty subset, while in *A* (containing at least two elements) there is not a single element that would not be connected by memory with at least one of the elements of *A*, then the *S-model* ^{pA} of the problem domain *pA* is an P-object that is defined by a pair of {$Mem^A \approx$ memory of the set of S-problems *A* of the problem domain *pA*}, {a family of *rul* (Mem^A) connections given on Mem^A}.

The non-empty set Mem^A of memory elements is divided into three subsets: the inputs Inp^A of S-problems, their outputs Out^A and a subset Or^A, each of whose elements is both the input and output of some S-problems. Any one of these subsets can be empty; Inp^A and Out^A can be empty at the same time.

Unlike the S-problem memory consisting of input and output, Mem^A contains a subset *Or* of memory elements, each of which can be either given (as input) or calculated (as output). The elements *Or* are called *reversible*, and *Or* is *a subset of reversible elements*. A subset *Inp* of Mem^A is called *a subset of given elements*, and a subset *Out* of Mem^A is *a subset of calculated elements*.

□ The S-problem domain *pA* is used to interpret the specifications of the desired S-problem objects represented in the *Lng* language. The interpretation consists of matching some subset (or a pair of subsets) of Mem^A to some subdomain of *pA*, called the *resolving S-structure*. □

□ *The S-problem graph* (*P-graph*) is a representation of the S–problem domain, designed for the implementation of *P-formalization* and *P-construction* of S-problem knowledge.

The set of *vertices of P-graph*, composed of S-problem objects, is called its *S-given basis* and is denoted by *P-basis*.

An edge of P-graph is a pair of vertices with a non–empty intersection in memory.

The P-graph edge load is determined by the set of all pairs of memory elements included in this intersection. *Vertex memory* Mem^{Vertex} is the memory of the S-problem (or S-problem domain) that the vertex represents. □

□ *A composite task Comp* is a subdomain of the S-problem domain *pA*, which contains at least two elements from the set *A* and on whose memory the partition is set: $Mem^{Comp} \approx Inp^{Comp} \cup Out^{Comp}$; $Inp^{Comp} \cap Out^{Comp} = ø$, which determines the input Inp^{Comp} and the output Out^{Comp} of *Comp*.

An oriented P-graph is assigned to *Comp*, the vertices of which are S-problems marked with their names. □

Depending on the composition of the vertices, the following *types of P-graphs* are defined:

- a *U-graph* has a set of vertices only from S-problems that are not represented by S-problem constructions;
- in a *C-graph*, at least one vertex is represented by a composite S-problem and there are no vertices represented by S-problem domains;
- in a *G-graph*, there is at least one vertex is represented by an S-problem domain (the rest may be composite S-problems and/or S-problems not represented by S-problem constructs).

IV. Resolving Structures on P-raphs

G-graphs serve as a means of formalizing knowledge about P-objects. ◊ The possibility of the existence of one or more vertices in the P-graph, which are S-problem domains, is of fundamental importance for the formalization of S-problem knowledge. ◊

The desired S-problem construction is given by a specification containing a description of memory and restrictions on the number of S-problem vertices (and, if necessary, restrictions related to the size of the S-problem, the accuracy of the result, etc.). The specification is interpreted on a P-graph, which serves as a representation of the S-problem domain of interest to the constructor. The construction mechanism on the P-graph serves as a means of interpreting specifications. Interpretation on a U-graph consists in matching a subset (or a pair of subsets) of U-graph memory elements with a subgraph whose memory would be in a given relation to the introduced subset (or a pair of subsets). The interpretations on the C-graph and G-graph are similar to the interpretations on the U-graph.

□ *An S-problem T is representable on a P-graph* if its input Inp^T is contained in a subset of $Giv^{P\text{-}graph} \cup Or^{P\text{-}graph}$, and the output Out^T is in a subset of $Comput^{P\text{-}graph} \cup Or^{P\text{-}graph}$ of P-graph memory. At the same time, there is at least one S-problem from the P-graph basis whose input is contained in or coincides with Inp^T. □

□ *The resolving structure* $Solv^T$ on P-graph, set in accordance with some problem *T*, is called a subgraph of P-graph with the minimum number of S-problem vertices on which the problem *T* is representable. □

□ *Interpretation of the S-problem vertex of the U-graph (or C-graph) in the process of searching for a resolving structure* consists in correlating the signification of the input and output elements of the S-problem. □

Rules for interpreting the S-problem vertex:

- if the input is fully signified, then the output is fully signified;
- if at least one element of the output is assumed to be signified, then the input is fully assumed to be signified.

The mechanism for constructing resolving structures conforms to the specification of the original S-problem subgraph on the P-graph by implementing three types of behavior in accordance with three types of requests:

1. For given subsets X and Y ($X \cap Y = \varnothing$) of the $Mem^{\text{P-graph}}$ memory of P-graph, then there exists a resolving structure $Solv^{XY}$ (with a minimum number of S-problem vertices, whose input is defined by X and output by Y) when there is a subgraph G whose set of vertices includes at least one vertex with a solvable S-problem, and the union of the outputs of the vertices of the subgraph G contains a subset of Y (or coincides with it).
2. For a subset X given on $Mem^{\text{P-graph}}$ memory of P-graph, then there is a $Solv^{X}$ resolving structure whose input is defined by a subset of X, and the output is a non-empty subset of $Mem^{\text{P-graph}}$, including the maximum number of elements that can be defined for a given X, when $X \cap Comput = \varnothing$ ($Comput < Mem^{\text{P-graph}}$) and there will be at least one vertex with a solvable S-problem.
3. For a subset Y given on $Mem^{\text{P-graph}}$, then there is a $Solv^{Y}$ resolving structure (with a minimum number of S-problem vertices, the output of which contains Y, and the input is composed of elements belonging to Giv) when $Y \cap Giv = \varnothing$.

For each of the three types of requests, a constructive proof of the existence of a resolving structure of the corresponding type is obtained.

After the resolving structure is found, the process of its *concretization* becomes feasible [21] in accordance with t*he specification of the conditions for the application* of the source S-problem.

V. Classes of Basic S-problems

S-representation. Creating of interconnected systems of S-(symbols, codes, signals), specification and programming languages, queries languages; representation of S-(messages, data), S-systems of notions and knowledge, etc. [4-22].

S-transformation. Converting S-messages (∘ speech ↔ text; analog ↔ digital; uncompressed ↔ compressed; *.odt ↔ *.pdf, etc. ∘) [2-7].

S-recognition. A necessary but insufficient condition for recognition is the presentation of S-message in a format known to recipient. When this condition is met, the tasks of matching the sample models or matching the properties of the recognized model with the properties of sample models are solved [2-7].

S-constructing. Constructing new S-objects from previously created S-objects (∘ specifications S-problems, S-(algorithms and programs), S-systems of notions and knowledge, etc. ∘). ◊ Elements of an S-construction are represented as *constructive S-objects* [2-7]. ◊

S-interpretation. Interpretation supposes the existence of an accepted S-message, an S-system of notions on which it should be interpreted, and an interpretation mechanism. ∘ To interpret a web page that is presented on a monitor screen, a person uses systems of notions stored in his memory. For an S-machine microprocessor, interpreted S-messages are codes of the S-machine commands and data that are represented by S-signals. For an S-machine compiler, interpreted S-message is an S-code of source S-program [2-7]. ∘

S-interaction. In this class, the problems of interaction in S-environment (*man – S-machine*; *S-machine – S-machine*) are studied. Senders and recipients of S-messages, a means of sending, transmitting and receiving messages are classified. Systems of S-messaging rules (S-network protocols), S-network architectures, service-oriented architectures, document management systems are being developed [3-24].

S-(*saving, accumulating and searching*). This class includes the related S-problems of saving, accumulating, and searching for S-messages. The S-machine memory devices, their management mechanisms, forms of storage and accumulation, methods of accumulation and search, databases and program libraries are studied [2-7].

S-security. S-problems of this class are designed to prevent and to detect vulnerabilities, to perform access control, to protect against unauthorized use, malware, and message interception [4, 25-26].

VI. Conclusion

The theory of S-symbols[6] is regarded as an integral component of the methodological framework for the advancement of artificial intelligence systems. This article covers a segment of the theory focused on the conceptual foundations of the theory, formalization of knowledge concerning S-problems and the classes of basic S-problems [3–4]. Some definitions have been clarified and examples have been added.

References

[1] Newell, A., Simon, H. 1976. Computer science as empirical inquiry: symbols and search. Commun. ACM **19**(3), 113–126. https://doi.org/10.1145/360018.360022.

[2] Berners-Lee, T. 2010. Long live the Web. Sci. Am. **303**(6), 80–85.

[3] Ilyin, V.D. 2023. Teoriya S-simvolov: kontseptual'nyye osnovaniya [The theory of S-symbols: Conceptual foundations]. Sistemy i sredstva informatiki [Systems and Means of Informatics] **33**(1), 126–134. doi: https://doi.org/10.14357/08696527230112.

[4] Ilyin, V.D. 2023. Teoriya S-simvolov: formalizatsiya znaniy ob S-zadachakh [The theory of S-symbols: Formalization of knowledge about S-problems]. Sistemy i sredstva informatiki [Systems and Means of Informatics] **33**(2), 24–131. doi: https://doi.org/10.14357/08696527230212.

[5] Ilyin, V.D. 2022. Symbolic Modeling (S-Modeling): an Introduction to Theory. In: Silhavy, R. (eds.) Artificial Intelligence Trends

[6] The theory of S-symbols was developed in the performance of research work N AAAA-A19-119091990038-2 ("Mathematical methods of data analysis and forecasting") under the state task of FANO of Russia for the Federal research center "Informatics and control" of the Russian Academy of Sciences.

in Systems. CSOC 2022. LNNS, vol. 502, pp. 585–591. Springer, Cham. Doi: 10.1007/978-3-031-09076-9_54.

[6] Rojek, I., Mikołajewski, D., Dostatni, E. 2020. Digital twins in product lifecycle for sustainability in manufacturing and maintenance. Appl. Sci. **11**(1), 31 pp.. doi: 10.3390/app11010031.

[7] Semeraro, C., Lezoche, M., Panetto, H., Dassisti, M. 2021. Digital twin paradigm: A systematic literature review. Comput. Ind. Vol. 130, art. 103469, 23 pp. doi: 10.1016/j.compind.2021.103469.

[8] Nguyen, H., Trestian, R., To, D., Tatipamula, M. 2021. Digital twin for 5G and beyond. IEEE Commun. Mag. **59**(2), 10–15. doi: 10.1109/MCOM.001.2000343.

[9] Jia, P., Wang, X., Shen, X. 2021. Digital-twin-enabled intelligent distributed clock synchronization in industrial IoT systems. IEEE Internet Things **8**(6), 4548–4559. doi: 10.1109/JIOT.2020.3029131.

[10] Ball, P., Badakhshan, E. 2021. Sustainable Manufacturing Digital Twins: A Review of Development and Application. In: Scholz, S.G., Howlett, R.J., Setchi R. (eds.) Sustainable Design and Manufacturing. KES-SDM 2021. Smart Innovation, Systems and Technologies, vol. 262. Springer, Singapore. doi :10.1007/978-981-16-6128-0_16.

[11] Liu, K., Song, L., Han, W., Cui, Y., Wang, Y. 2022. Time-varying error prediction and compensation for movement axis of CNC machine tool based on digital twin. IEEE Trans. Ind. Inf. **18**(1), 109–118. https://doi.org/10.1109/TII.2021.3073649.

[12] Rathore, M., Shah, S., et al. 2021. The Role of AI, Machine Learning, and Big Data in Digital Twinning: a Systematic Literature Review, Challenges, and Opportunities. IEEE Access 9, 32030–32052. https://doi.org/10.1109/ACCESS.2021.3060863.

[13] Kim, R.Y. 2011. Efficient Wireless Communications Schemes for Machine to Machine Communications. In: Zain, J.M., Wan Mohd, W.M.b., El-Qawasmeh, E. (eds.) Software Engineering and Computer Systems. ICSECS 2011. Communications in Computer and Information Science, vol. 181. Springer, Berlin, Heidelberg. Doi: 10.1007/978-3-642-22203-0_28.

[14] Lien, S.Y., Liau, T.H., et al. 2012. Cooperative access class barring for machine-to-machine communications. IEEE T. Wirel. Commun. **11**(1), 27–32. doi: 10.1109/TWC.2011.111611.110350.

[15] Wei, Y., Blake, M.B. 2010. Service-oriented computing and cloud computing: Challenges and opportunities. IEEE Internet Comput. **14**(6), 72–75. doi: 10.1109/MIC.2010.147.

[16] Perera, C., Liu, C.H., Jayawardena, S. 2015. The emerging Internet of Things marketplace from an industrial perspective: A survey. IEEE T. Emerging Topics Computing **3**(4), 585–598. doi: 10.1109/TETC.2015.2390034.

[17] Kakhno, M.I., Kal'ya, A.P., Tyugu, E.Kh. 1988. Instrumental'naya sistema programmirovaniya dlya ES EVM [The Instrumental Programming System for ES EVM]. Finansy i statistika, Moscow.

[18] Manna, Z., Waldinger, R. 1975. Knowledge and Reasoning in Program Synthesis. Artificial Intelligence **6**(2), 175–208. doi: 10.1016/0004-3702(75)90008-9.

[19] Sheu, P.CY. 1997. Automatic Program Synthesis and Reuse. In: Software Engineering and Environment. Software Science and Engineering. Springer, Boston, MA. Doi: 10.1007/978-1-4615-5907-8_8.

[20] Basin, D., Deville, Y., Flener, P., et al. 2004. Synthesis of Programs in Computational Logic. In: Bruynooghe, M., Lau, KK. (eds.) Program Development in Computational Logic. LNCS, vol. 3049. Springer, Berlin, Heidelberg. Doi: 10.1007/978-3-540-25951-0_2.

[21] Ilyin, V.D. 1989. Sistema porozhdeniya programm [Program generating system]. Nauka, Moscow. ISBN: 5-02-006578-1.

[22] Ilyin, V.D. 1995. A methodology for knowledge based engineering of parallel program systems. In: Forsyth, G.F., Ali, M. (eds.) Proceedings of the Eighth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Melbourne, Australia, Jun 06-08, 1995 (IEA/AIE-95). pp. 805–809. Gordon and Breach Publishers, Melbourne.

[23] Ilyin, A.V. 2007. Konstruirovaniye razreshayushchikh struktur na zadachnykh grafakh sistemy znaniy o programmiruyemykh zadachakh [Construction of resolving structures on problem graphs of the knowledge system about programmable tasks]. Informatsionnyye tekhnologii i vychislitel'nyye sistemy [Journal of Information Technologies and Computing Systems] 3, 30–36.

[24] Ilyin, A.V., Ilyin, V.D. 2021. Updated methodology for task knowledge based development of parallel programs. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) CoMeSySo 2021. LNNS, vol. 231, pp. 319–328. Springer, Cham. Doi: 10.1007/978-3-030-90321-3_25.

[25] Ilyin, A.V., Ilyin, V.D. 2016. Variational online budgeting taking into account the priorities of expense items. Agris On-line Papers in Economics and Informatics **8**(3), 51–56. doi: 10.7160/aol.2016.080305.

[26] Ilyin, A.V., Ilyin, V.D. 2018. Situational management: review of the results relevant to the development of online services for e-government and e-business. Journal of information technologies and computing systems 4, 45–54. doi: 10.14357/20718632180405.

Vladimir D. Ilyin. Doctor of Science in technology, Professor, Federal Research Center «Computer Science and Control» of the Russian Academy of Sciences, 40 Vavilova str., Moscow,119333, Russia, e-mail: vdilyin@yandex.ru. ORCID: https://orcid.org/0000-0002-9761-082X.