Computer Science and Information Technology 1(2): 125-131, 2013 http://www.hrpub.org
DOI: 10.13189/csit.2013.010208

Generation of Questions Sequences in Intelligent Teaching Systems Based on Algebraic Approach

Alexander Zuenko[1], Alexander Fridman[1,*], Boris Kulik[2]

[1]Institute for Informatics and Mathematical Modeling, Kola Science Centre of the Russian Academy of Sciences (RAS), 24A Fersman str., 184209, Apatity, Russia
[2]Institute of Problems in Machine Science of the RAS, 61 Bol'shoi pr., 199178 St. Petersburg Russia
*Corresponding Author: fridman@iimm.kolasc.net.ru

Abstract The paper describes an approach to development of question-and-answer teaching systems based on controlled languages and algebraic models for representation and processing of question-and-answer texts. We propose using a partial order relation "question-subquestion" to build an individual trajectory of teaching. To model a strategy of examination, we use defeasible reasoning formalized within our earlier developed *QC*-structures.

Keywords Intelligent Teaching System, Question-and-answer Text, *QC*-structure, *N*-tuple Algebra

1. Introduction

There are four main parts in intelligent teaching systems (ITS), namely teaching material, a teaching unit, a knowledge control unit and a check unit [1]. Here we consider the last two units only. The knowledge control unit analyses how a student has learned the material, the check unit evaluates knowledge of a student and puts a mark. Separating the knowledge control unit from the check unit allows determining the part of knowledge that the student lacks. In some ITSs, the knowledge control unit is included either in the teaching unit or in the check unit.

Papers dealing with ITSs mostly introduce ideas how check systems can determine abilities of a student by analyzing his decisions [2]. Such systems focus on putting a mark.

Unlike these systems, when a teacher gets a wrong answer, he/she tries to reveal vacancies in the student's knowledge by putting leading questions or additional tasks. As a result, the teacher either "leads" the student to the right answer or determines the concept, rule or theorem, which the student does not know or cannot use.

In the authors' opinion, an ITS should work more as a real teacher. Correspondingly, the ITS is not to only estimate the degree of knowledge for a student, but also to reveal poorly perceived knowledge and to advise ways for its improving.

An ITS controls a question-and-answer dialog to appraise student's knowledge. The unit simulating teacher's examination strategies plays an important role in this dialog. It also provides revision of estimates of student's knowledge accumulated during examination.

The situation becomes more complicated when this dialog stipulates natural language communication with "arbitrary" shapes of answers. The term "arbitrary" is not quite correct as the student is supposed to be familiar with the teaching material i.e. immersed into the context. This means the student ought to give reasonable answers within the terminology relevant to the teaching material.

Lately, CNLs (Controlled Natural Languages) are mostly popular for development of dialog systems admitting arbitrary answers of users. A CNL is a version of a natural language simplified by means of a problem-methodological context and created for solving of certain tasks [3]. An original variant of a CNL for question-and-answer dialogs controlled by conceptual grammars was proposed in [4]. A semantical classification of question-and-answer texts was given there as well.

In this paper, we propose an approach to building ITSs based on an algebraic interpretation of the mentioned model of the question-and-answer dialog. We use our n-tuple algebra (NTA) [5-7] and an extension of partially ordered sets, namely QC-structures [8], to represent and analyze question-and-answer texts.

Let us now briefly describe NTA [5-7].

2. Basics of N-tuple Algebra

N-tuple algebra was developed for modeling and analyzing of *n*-ary relations. Unlike relational algebra that is used for formalization of databases, NTA can use all laws and techniques of mathematical logic for logical modeling and analysis of systems, namely logical inference, corollary verification, analysis of hypotheses, abductive inference, etc.

NTA is an algebra of *n*-ary relations based on features of

Cartesian products.

In NTA, we represent relations as four types of structures described below and called *NTA objects*. Every NTA object is immersed into a certain space of attributes. Names of NTA objects contain an identifier followed by a sequence of attributes names in square brackets; these attributes determine the relation diagram in which the NTA object is defined. For example, R[XYZ] denotes an NTA object defined within the space of attributes X, Y and Z.

NTA objects provide a condensed representation of *n*-ary relations. When necessary, by means of specific algorithms the objects can be transformed into ordinary *n*-ary relations containing sets of *n*-tuples called *elementary n-tuples* in NTA. The Cartesian product of domains for a given sequence of attributes forming the relation diagram of an NTA object is called a *partial universe*.

Structures defined on the same relation diagram are called *homotypic* ones. Any collection of homotypic NTA objects is an algebra of sets. In NTA, it is possible to implement operations of algebra of sets on NTA objects with different relation diagrams.

NTA objects, namely *C*-n-tuples, *C*-systems, *D*-n-tuples, and *D*-systems are formed as matrices of subsets of attributes domains called *components*. The components include two types of *dummy components*. One of them is called the *complete component*; it is used in *C*-n-tuples and denoted by "*". A dummy component "∗" added in the *i*-th place in a *C*-n-tuple or in a *C*-system equals to the set corresponding to the whole domain of the *i*-th attribute in the relation diagram. Another dummy component (∅) called an *empty set* is used in *D*-n-tuples.

Let us now proceed with description of main NTA structures; they are *C*-systems and *D*-systems.

We record a *C*-system as a matrix of component sets framed with brackets.

C-systems are convenient for representing disjunctive normal forms (DNFs) of finite predicates. A one-line *C*-system is called a *C-n-tuple*; it is similar to a row vector in matrix algebra. In logic, a *C*-n-tuple corresponds to a separate conjunct.

D-systems model conjunctive normal forms (CNFs) of finite predicates. We denote a *D*-system as a matrix of component sets framed with reversed brackets. *D*-systems provide easy calculating of *C*-systems' complements.

A one-line *D*-system is called a *D-n-tuple*. In logic, a *D*-n-tuple corresponds to a separate disjunct.

Calculations of unions and intersections for *C*- and *D*-structures are specific; you can find their description in [2,3].

Please note that NTA provides implementing all operations of algebra of sets and all checks of relations among NTA objects (e.g., equality and inclusion) in matrix form, without having to represent these objects as sets of elementary *n*-tuples. Absence or presence of a given elementary *n*-tuple in any NTA object can be checked by a gorithms with polynomial complexity [2].

To process NTA objects defined on different diagrams, we have developed some operations on attributes, addition of a dummy attribute (+*Attr*) and elimination of an attribute (-*Attr*) in particular. The operation +*Attr* corresponds to the rule of generalization in predicate calculus, so it does not change the semantics of any relations. This operation upon any NTA object simultaneously adds the name of a new attribute into the relation diagram and adds a new column with dummy components into the corresponding place of a matrix representation.

The operation +*Attr* is often used to reduce some different-type NTA objects to the same relation diagram. Then we can perform all necessary operations and checks by means of standard NTA algorithms. Considering this, we have introduced *generalized operations* ($\cap_G$, $\cup_G$ and calculating a complement). These operations completely correspond to logical connectives: conjunction, disjunction and negation. Our algebra of relations with these generalized operations is proved to be isomorphic to the ordinary algebra of sets. This way we have eliminated the restriction in the theory of relations stating that algebra-of-sets laws are only applicable to the relations defined upon the same Cartesian product.

Suppose that we have a system of premises (axioms) A_1, …, A_n represented as NTA objects. NTA provides solving the following problems of deductive analysis [2,3].

1) *Problem of correctness check for an alleged consequence from the axioms.*

2) *Problem of derivation of possible consequences* considering certain semantical constraints, for instance, presence of given variables or their combinations in a consequence, deriving a consequence with the minimal number of significant variables, etc.

Unlike other logical systems that solve such problems by using inference rules with hardly optimized order of their implementation, NTA solves these problems by means of certain standard algorithms. To do so, we transform classical logic' formulas expressing premises and consequences into NTA objects and subject these objects to generalized operations and checks of equality ($=_G$) or inclusion ($\subseteq_G$). The transition to the algebraic representation becomes clearer, if we consider that NTA objects model scope of truth for logical formulas. Then the correctness proof for an alleged consequence B from the premises A_i (the consequence and the premises can be different-type NTA objects) requires for calculation of generalized intersections and check the following generalized inclusion:

$$(A_1 \cap_G \ldots \cap_G A_n) \subseteq_G B. \quad (1)$$

This relation allows checking correctness not for the inference rules of classical logic only, but also for rules specific to a certain knowledge system.

All logical inference rules in mathematical logic and the natural deduction calculus by G. Gentzen [4] are proved to satisfy the relation (1) for the consequence and the premises transformed into NTA objects.

To derive possible consequences from the given premises according to (1), we calculate an NTA object $A = A_1 \cap_G \ldots$

$\cap_G A_n$ first, then we choose a B_i for which $A \subseteq_G B_i$ is true. This means any correct consequence modeled by an NTA object is a superset of A. There are several techniques to build such a consequence. Here we describe some of them only.

1) *Calculation of projections* of the NTA object A is the best to perform when A is a C-system. Then B_i can be obtained by elimination of attributes from A. This way we can form consequences with a given set of attributes.

2) If A is a D-system, two techniques are applicable. The first one requires for transformation A into a C-system followed with forming consequences according to the previous technique. Here is a disadvantage because the transformation algorithm from a D-system into a C-system has exponential complexity in general case. So, it needs essential computational resources, if A is of large dimension.

3) For A expressed as a D-system, there is a second (polynomial in complexity) technique to derive consequences by deleting some rows from the matrix representation of A. The resulted B_i always obey the relation $A \subseteq B_i$, but it is not easy to obtain consequences with a given set of attributes this way.

The described inference system simplifies building consequences with a given set of attributes (variables) or proving their absence. Solving this task with usage inference rules is only possible by the exhaustive search of variants.

Now let us consider some details of building ITSs within frames of the CNL concept.

3. Building a Question-and-answer Dialog Based on a CNL

During a question-and-answer dialog when the ITS plays the active part, the context of the dialog determines the range of expectable and possible answers fairly clear. Sense typification of questions and semantical classification of response texts provides assigning every type of questions with a limited set of admissible (i.e. logically correct) sense constructs (response formulas). As a result, the content of an answer, its dictionary, shape and partly volume are predefined, and a user has to answer a question within definite frames [9]. Indeed, if an ITS forms questions basing on lectures material familiar to a trainee, we can even define the sets of expectable lexical units for an answer to a given question. Such approach makes it possible to create an effective system checking correctness of an answer to this question.

The NL-texts interpretation system for an ITS-driven context [9] contains a knowledge base, a base of specific grammar constructs, a lexical processor and a semantical interpreter. The answer text recognition includes translation the text into a canonical representation, search for a semantical pattern of the reference answer and collation of the answer with this pattern.

Eventually, the ITS forms a certain situational vector consisting of a set of indices of the answer quality. These indices determine the further dialog.

The ITS knowledge base is a number of models of teaching texts, sets of testing questions and reference answers. These models are created by a subject domain expert in computer-aided mode. The classification of questions developed in [10] helps to build templates of answers. Questions are split into five types as follows.

Questions of the first type require for an explicit definition of some key parameters without considering relations among them. The second-type questions demand a user to point one relation corresponding to one major concept. When answering a third-type question, a user has to notice a composition of a certain set of basic relations linked with a major concept. An answer to a forth-type question has to include an arbitrary composition of relations assigned to one concept. To answer a fifth-type question, a user shall define several concepts interconnected with an arbitrary set of relations.

Splitting texts into semantical classes is based on revealing a major concept (or several major concepts) and relations connected with this concept. By using some definite characteristics, we can classify the set of specific concepts and relations into a finite number of concepts types and relations types. According to terminology [9], we call them semantical units or conceptulas. If we assign every concept and relation with a certain type, we can translate every sensible clause from a subject domain into a text composed of the concepts types and relations types (i.e. semantical units) without any detailed consideration of grammar features of the lexemes. The semantical classes of answers introduced above correspond to some patterns of conceptulas combinations, which render a specific generalized sense of answers to a given class of questions (meanings of questions). Individual conceptual grammars (ICGs) are such patterns of conceptulas combinations, which render an expected sense properly.

In the given paper, we focus on ways to build patterns (templates) of an answer.

The pattern of an answer is built based on the question to be asked. The pattern is a twain <F, G> where G denotes the ICS for the class of answers corresponding to a given question. F = <L, K> is an informational structure containing lexemes L, which represent concepts and relations and their reference roles K in an answer.

As an illustrative example, let us consider the following teaching text from a lecture [10].

"A compiler is a program situated in the random access memory (RAM) that translates a source text written in a high-level programming language (HLPL) into an object text in a machine-language code (MLC). Compilation includes syntactical and lexical analyses, codes generation and optimization. Compilation takes place before loading that transforms the object module into a run-time module and loads it into the memory. The linker writes the run-time module on a disk".

Let a third-type question is asked, namely "What is the function of a compiler?". Some possible answers are as

follows [10].

1. translates a source text written in a high-level programming language into an object text in a machine-language code,
2. makes MLC from HLPL,
3. translates HLPL into MLC.

All listed answers belong to the FUNCTION class. To describe grammars of answers to this class, we use the following conceptulas [10]:

SS is a conceptula representing a major concept; ***SA*** represents an argument concept, SP represents a result concept; ***GPA*** is a preposition before ***SA***; ***GPP*** is a preposition before ***SP***; ***RA*** is a conceptula reflecting the relation of ***SS*** to ***SA***; ***RP*** is a conceptula reflecting the relation of ***SS*** to ***SP.***

A formalized representation of the listed answers looks like this:

1) $\boldsymbol{R_A \rightarrow SA \rightarrow GP_P \rightarrow SP}$

2) $R_P \rightarrow SP \rightarrow GP_A \rightarrow SA$

3) $\boldsymbol{SS} \rightarrow R_A \rightarrow SA \rightarrow GP_P \rightarrow SP$

Here $\boldsymbol{R_A}$ stands for the relation "translates"; $\boldsymbol{R_P}$ denotes the relation "makes"; ***SA*** contains concepts "source text written in a high-level programming language", "HLPL"; ***SP*** reflects concepts "object text in a machine-language code", "MLC"; the preposition "from" is in $\boldsymbol{GP_A}$; the preposition "into" is in $\boldsymbol{GP_P}$, and the concept "compiler" is in ***SS***. The arrows display the sequence chain for conceptulas in the sentence.

Similar formulas representing the sequence and roles of lexemes form an ICG. The structure ***F*** looks like

ANSWER: THE CLASS = FUNCTION

F: ***SS*** = *compiler, translator*; $\boldsymbol{R_A}$ = *translates, transforms*; $\boldsymbol{R_P}$ = *receives*; ***SA*** = *source text written in a high-level programming language, text in a high-level programming language, HLPL*; ***SP*** = *object text in a machine-language code*, *text in a machine-language code, MLC program, MLC*.

This listing contains lexemes (***L***), "*compiler*" for instance, and their semantical roles (***K***) in a sentence, the conceptula "***SS***" as an example.

For convenience, we will write such structures as Table 1 below.

Table 1. Example of an informational structure ***F***

ANSWER: CLASS = **FUNCTION**{

SS	RA	RP	SA	SP
{compiler, translator}	{translates, transforms}	{makes, receives}	{source text written in a high-level programming language, text in a high-level programming language, HLPL}	{object text in a machine-language code, text in a machine-language code, MLC program, MLC}

}// END OF ANSWER CLASS = **FUNCTION**

In this case, we can represent the informational structure ***F*** as a Cartesian product of conceptulas meanings' sets. In a more general case, when we need to consider some correspondences among lexemes, we can write the informational structure ***F*** as a system of *n*-ary relations.

To model and process typical structures needed in question-and-answer components of an ITG, we propose to use *n*-tuple algebra [7,11,12]. In particular, it is possible to concisely write templates of answers as *C*-systems.

Further on, we describe how to solve the automation problem for checking knowledge of a trainee.

4. An Algebraic Model of a Question-and-answer Dialog

A knowledge control unit of an ITS belongs to open systems able to replenish their knowledge and previous conclusions in order to reflect a real state of learning when the situation changes. To formalize reasoning in such systems, argumentation can be applied.

When examining a student, a teacher usually has a certain sequence of questions to ask. If an answer is wrong, some subquestions come to assist in revealing vacancies in the student's knowledge. Examination lasts as long as the teacher gets enough information to ground a decision regarding the knowledge level of a trainee and/or proposals for additional study of the teaching material.

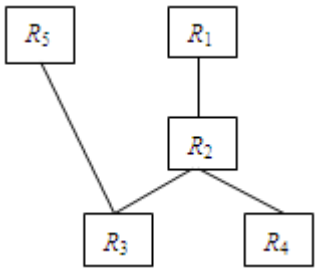

Figure 1. A chart of a questions set

Let us define a subordination relation S upon the set of questions that means as follows. An x is subordinated to an y, if it is necessary to know a right answer to the question x in order to answer the question y properly. Suppose we have the following questions with a corresponding subordination relation (see Fig. 1).

R1. What do you know about compilers and compiling? (A 5-th type question)

R2. Give a definition of a compiler (a 4-th type question)

R3. Where a compiler is situated? (a 2-nd type question)

R4. What is the function of a compiler? (a 3-rd type question)

R5. What do we use PC RAM for? (a 3-rd type question)

Fig. 1 shows that questions of a more complex type dominate questions with simpler types (if they concern the same major concept). For instance, the 4-th type question R2 dominates the 3-rd type question R4. In other words, the proposed classification of questions provides for a natural ordering of the questions set assuming that answers will contain common concepts and relations. According to Suleymanov[9], we can correlate every question with an answer pattern. Below we will dwell on the informational structure **F** of this pattern. Answer patterns for subquestions contain only a part of relations defined in their questions themselves. We can see this by comparison the questions R2 and R4.

We have already analyzed the answer pattern for the question R4 in the previous section. The question R2 can have the following answer pattern based on the source text given above (see Table 2).

Table 2. Answer Pattern for the Question *R*2

ANSWER: CLASS = **DEFINITION**{

SS (major concept)	S_{GC} (a more general concept than the major one)
{compiler, translator}	*{program}*

CLASS = FUNCTION

R_A	R_P	*SA*	*SP*
{translates, transforms}	*{makes, receives}*	*{source text written in a high-level programming language, text in a high-level programming language, HLPL}*	*{object text in a machine-language code, text in a machine-language code, MLC program, MLC}*

CLASS = ***SR*** (space relation)

SO	R_{SO}	R_{OS}
{random access memory, RAM}	*{is situated in, is in, contained in}*	*{contains}*

}// END OF ANSWER CLASS = **DEFINITION**

Join of all three relations recorded as tables above forms an NTA object [13,14] defining the answer pattern for the question R2.

The main argumentation procedure consists in defining the true statements with using their sets of arguments. The set of propositions and corresponding arguments can be defined as K = (V, A, I) where V is the set of propositions; A is the set of arguments; I is such a relation upon the set V × A that pIaσ where p ∈ V, aσ ∈ A; σ ∈ {+, –} is true if and only if aσ is an argument of the proposition p [1].

Below, we briefly describe implementation of ITSs in NTA.

As we have already mentioned, within classical logic where we deal with the "closed world model", NTA structures are good to represent the truth set of a logical formula. If the union of the truth set and the falsity set does not equal to a universe (this is the "open world model"), we use NTA objects to model corresponding sets of existence (set of positive examples) and sets of exemptions (sets of negative examples).

Usually, defeasible reasoning and argumentation systems (including the proposed system) are based on the "open world model" where subject domain axioms may be and have to be revised. In our case, these axioms state dependencies among questions denoted as material implications.

We use NTA structures to implement the following stages of the proposed defeasible reasoning:

1) correctness recognition of a trainee answer;

2)generation of new axioms based on the current information regarding the trainee knowledge.

During the first stage, a trainee answer formalized as a *C*-n-tuple is classified as a right one or as a wrong one. If this is not possible yet, the decision is made to continue examination. So, every following question is not fixed as in the most testing systems. Conversely, it is chosen depending on the previous answers of the trainee.

During the second stage, our ITS forms new axioms and rejects the unsupported ones using the current estimates of the previous answers. By using NTA techniques, we provide automatic generation of axioms denoted as material implications based on relations among the NTA objects, which model the answer as corresponding sets of existence and sets of exemptions.

The representation of axioms was detailed above. After their generation, the NTS concludes consequences and reveals different "collisions" [8] above formal contradictions. Here we use *QC*-structures [8] and two inference rules, namely contraposition and transitivity.

For building an argumentation logic, elementary propositions are the statements like "a student knows a proper answer to the given question". If we interpret right answers as arguments "pro" and wrong answers as arguments "contra", the relation S upon the questions set provides creating an argumentation logic with an ordered set of arguments [15].

Let the question Ri correlates with the proposition pi ("a student knows a proper answer to the given question") and the question Qj correlates with the proposition qj ("a student knows a proper answer to the question j"). According to the proposed segmentation of the question-and-answer text, the relation Ai ⊆G Aj is always true if the question Ri dominates the question Qj. If the relation Ai ⊆G Aj is true, we can state that V[(pi→qj)]=1 (V is the rating function).

If an answer pattern not only contains samples of proper answer, but also prohibits some answers or certain lexemes, every statement pi can correlate (above the NTA object Ai) with another NTA object Bi describing forbidden answers. According to terminology [16], the NTA object Ai defines the range of the function g+ for the proposition pi and the NTA object Bi defines the range of the function g–.

Analysis of arguments sets allows us to reveal some cause-and-effect relations within a subject domain. This possibility is grounded by the definition of implication in argumentation logic. The sufficient condition for implications to be true looks like follows [15].

If

$$g^{+}(p) \subseteq g^{+}(q) \text{ and } g^{-}(q) \subseteq g^{-}(p), \text{ then } V[(p \rightarrow q)] = 1;$$

$$g^{+}(p) \subseteq g^{-}(q) \text{ and } g^{+}(q) \subseteq g^{-}(p), \text{ then } V[(p \rightarrow \neg q)] = 1.$$

By analyzing the arguments sets, we can conclude two types of statements, namely p→q and p→¬q. If we accept the definition of collisions as a generalization of concepts "argument undercutting", "counter-evidence (attack)" [14], we can use QC-structures earlier developed by one of the authors [8], in order to detect and eliminate collisions in argumentation systems.

Let us describe one of the possible scenarios of the control process by means of introduced techniques of logical analysis for the case when checking starts from the most complex question R1 (see Fig. 1). Suppose a student answered the question R1 incorrectly. Then the system asks the question R2. If the student gives a wrong answer again, the system asks the question R3. If the answer is wrong once again, the system asks the question R4. If R4 is answered properly, the system asks the question R3 once more. If the answer is right this time, the question R2 is put again followed by the question R5. Provided all answers are proper, the question R1 is put again.

5. Conclusion

We propose algebraic models to represent and process question-and-answer texts intended for control of trainees' knowledge. The developed models are open for replenishing of knowledge. They support knowledge control and forming an individual teaching trajectory by means of argumentation procedures.

The described above approach to building question-and-answer systems based on Controlled Natural Languages stipulates representation and processing of information formalized as *n*-ary relations. Then NTA provides a more flexible representation for combinations of

possible/forbidden answers and significant complexity decrease for different stages of defeasible reasoning by means of compact notation of *n*-ary relations and usage of specific algorithms to accelerate computational procedures [5-7].

Acknowledgements

The authors would like to thank three institutions for their assistance in partial funding of this research: the Russian Foundation for Basic Researches (grants 13-07-00318, 12-07-00689, 12-07-000550, 12-07-00302, 11-08-00641), the Chair of the Russian Academy of Sciences (project 4.3 "Intelligent Databases" of the Programme # 16 of Basic Scientific Researches) and the Department for Nanotechnologies and Information Technologies of the Russian Academy of Sciences (project 2.3 within the current Programme of Basic Scientific Researches).

REFERENCES

[1] T.A. Taran, A.I. Rivkind. Argumentation System for Knowledge Control, News in Artificial Intelligence, No.5-6, 12-18, 2001. (in Russian).

[2] I.Yu. Denisova, P.P. Makarychev. Mathematical Models of Ontology for Database of Teaching Information system, Ontology of Designing, Vol.3, 62-78, 2012. (in Russian).

[3] J. Pool. Can controlled languages scale to the Web? Proc. of the 5th International Workshop on Controlled Language Applications (CLAW 2006), 2006.

[4] D.Sh. Suleymanov. Two-Level Semantic Processor of Natural-Language Answer Texts, Open Semantic Technologies for Intelligent Systems (OSTIS-2011): Proc. of International Scientific-Technical conf. (Minsk, 10-12 February 2011): Minsk BGUIR, 311-322, 2011. (in Russian).

[5] B.A. Kulik. A Logic Programming System Based on Cortege Algebra, Journal of Computer and Systems Sciences International, Vol.33, No.2, 159-170, 1995.

[6] A.A. Zuenko, B.A. Kulik, A.Ya. Fridman. Unified Processing of Data and Knowledge Based on General Theory of N-ary Relations, Artificial Intelligence and Decision Making, No.3, 52-62, 2010. (in Russian).

[7] B.Kulik, A. Zuenko, A. Fridman. An Algebraic Approach to Intelligent Processing of Data and Knowledge. Saint Petersburg Polytechnic University, 2010 (in Russian).

[8] B. Kulik. Logic of Natural Reasoning. Nevsky Dialekt, Saint Petersburg, 2001. (in Russian).

[9] D.Sh. Suleymanov. Two-Level Semantic Processor of Natural-Language Answer Texts, Open Semantic Technologies for Intelligent Systems (OSTIS-2011): Proc. on International Scientific-Technical conf. (Minsk, 10-12 February 2011): Minsk BGUIR, 311-322, 2011. (in Russian).

[10] D.Sh. Suleymanov. Systems and Information Technologies to Process Natural-Languages Texts Based on Pragmatically-O riented Linguistic Models. Thesis Dr. of Science (Techn.). Kazan, 2000. (in Russian).

[11] B. Kulik, A. Fridman, A. Zuenko. Logical Inference and Defeasible Reasoning in N-tuple Algebra. In: "Diagnostic Test Approaches to Machine Learning and Commonsense Reasoning Systems", IGI Global, 102-128, 2012.

[12] B. Kulik, A. Fridman, A. Zuenko. Logical Analysis of Intelligence Systems by Algebraic Method, Cybernetics and Systems 2010: Proc. of Twentieth European Meeting on Cybernetics and Systems Research (EMCSR 2010), Vienna, Austria, 198-203, 2010.

[13] M.M. Ayupov, B.A. Kulik, O.A. Nevzorova, D.Sh. Suleymanov, A.Ya. Fridman. Approach to Building Question-and-Answer Systems Based on Networks of N-ary Relations, Proc. of the 13-th National conf. on Artificial Intelligence with International Participation (16-20 October 2012, Belgorod, Russia): Belgorod University Vol. 2, 152-160, 2012. (in Russian).

[14] A.A. Zuenko, B.A. Kulik, A.Ya. Fridman. Integration of Data and Knowledge Bases within Algebraic Approach, Open Semantic Technologies for Intelligent Systems (OSTIS-2011): Proc. on International Scientific-Technical conf. (Minsk, 10-12 February 2011): Minsk BGUIR, 59-70, 2011. (in Russian).

[15] V.N.Vagin, E.Yu. Golovina, A.A. Zagoryanskaya, M.V. Fomina. Exact and Plausible Reasoning in Intelligent Systems, Fizmatlit, Moscow, 2008. (in Russian).

[16] V.K. Finn. On One Variant of Argumentation Logic , NTI, Series 2, No.5-6, 3-19, 1996. (in Russian).