Theory of K-representations as a Tool for Designing File Managers with a Natural Language Interface

V.A. Fomichov* and A.A. Razorenov**

* Department of Innovations and Business in the Sphere of Informational Technologies, School of Business Informatics, Faculty of Business and Management, National Research University Higher School of Economics, Kirpichnaya str. 33, 105187 Moscow, Russia

** Technoserv Consulting, 115280 Moscow, Russia

vfomichov@hse.ru

***Abstract* - During last decade, semantic parsing of the instructions in natural language (NL) has become a significant branch of the studies aimed at creating semantics-oriented NL processing systems. A topical problem of the kind is designing file managers with a NL-interface. The principal attention in our previous papers was paid to creating new, more effective theoretical foundations of designing semantic parsers of NL-instructions. In particular, we suggested in a very concise form an original and broadly applicable algorithm of semantic parsing. This paper, firstly, illustrates a correspondence between input instructions and their semantic representations. Secondly, the main attention is given to describing mathematical foundations of executing NL-instructions by computer systems. The methodological basis for these results is the theory of K-representations (knowledge representations). Its basic formal model introduces a system consisting of ten partial operations on conceptual structures. There are solid grounds for conjecturing that, combining these operations in arbitrary order, it is possible and convenient to build step by step a semantic representation of arbitrarily complex sentence or discourse in NL. The stated theoretical results have become the basis for designing a file manager with a NL-interface NLC-2 (Natural Language Commander - Version Two).**

Keywords* - *semantic parsing; SK-language; integral formal semantics of natural language; execution of instruction; relation of correspondence to a pattern

I. INTRODUCTION

Semantic parsing of the instructions in natural language (NL) has become during last decade a significant branch of the studies aimed at designing semantics-oriented NL processing systems. This branch is interested, in particular, in the development of NL-interfaces for interaction with robots and the personages of videogames [1-4], navigation in virtual space [5, 6], and for programming by means of NL-instructions [7, 8]. Besides, one of the topical problems of the kind is the development of file managers with a NL-interface. This problem is considered by us in several previous papers, in particular, in [9-13]. The principal attention was paid to creating new, more effective theoretical foundations of designing semantic parsers of NL-instructions.

In [11, 12], an original and broadly applicable algorithm of semantic parsing SemSyntRA is presented in a very concise form. This paper, firstly, illustrates a correspondence between input NL instructions and their semantic representations (SRs). Secondly, the main attention is given to describing mathematical foundations of executing NL-instructions by computer systems. The methodological basis for these results is provided by the theory of K-representations (knowledge representations), or TKR [14-22]. It is the central constituent of Integral Formal Semantics of NL (see [23] and Chapter 2 of [19]).

The basic mathematical model of TKR introduces a system consisting of ten partial operations on conceptual structures. There are solid grounds for conjecturing that, combining these operations in arbitrary order, it is possible and convenient to build step by step a semantic representation (SR) of arbitrarily complex sentence or discourse in NL (English, Croatian, Russian, etc.).

The basic model of TKR determines a new class of formal languages - the class of SK-languages (standard knowledge languages). The SRs of NL-texts are to be the expressions of SK-languages.

The stated theoretical results have become the basis for designing a file manager with a NL-interface NLC-2 (Natural Language Commander - Version Two). This system is implemented with the help of the functional programming language Haskell.

The structure of this paper is as follows. Section 1 contains an introduction to the considered problem. Section 2 gives a very general information about TKR as a whole and about its basic mathematical model determining, in particular, the class of SK-languages. Besides, this section illustrates the correspondence between an input NL instruction and its primary SR generated by the algorithm SemSyntRA and being an expression of a certain SK-language. Section 3 explains the notion of a transformation rule used for constructing a secondary SR of an input instruction from its primary SR. The application of the transformation rules depends on special binary relation on the considered SK-language, it is called the relation of correspondence to a pattern. This relation is explained in Section 4. Section 5 shortly characterizes an applied intelligent system called NLC-2

Natural Language Commander - Version 2). The ideas described in this paper underpinned the design of this system being a file manager with a NL-interface. Section 6 contains the conclusions.

II. SK-LANGUAGES AS A CONVENIENT TOOL FOR DESCRIBING COMPLEX SEMANTIC STRUCTURES OF NATURAL LANGUAGE

One of the principal reasons for developing the algorithm SemSyntRA was the intention to considerably expand the possible complexity of input NL instructions. The algorithms of NL instructions' semantic parsing described in scientific literature by the middle of this decade are able to process only simple instructions, including one verb with dependent words. But the instructions emerging in applications may be much more complex: include several actions joined by the connective AND or OR, indicate the order of actions, time distance between actions, mention compound designations of objects' groups as the operands of actions, include the modal words "necessary", "should", etc.

The analysis of the scientific literature shows that the main approaches to this problem used in practice are Abstract Meaning Representation (AMR) [3] and lambda-calculus meaning representation (LCMR) [1].

The semantic formalism AMR was introduced in 2013 in the ACL publication [24] by a group consisting of ten researchers from UK and USA. The paper [25] shows that much broader prospects for creating semantic languages-intermediaries in comparison with AMR are opened by the theory of K-representations, developed by V. A. Fomichov [14-22]. The advantages of TKR in comparison with AMR are, in particular, the possibilities to construct semantic representations of compound infinitive constructions (expressing goals, commitments, etc), of compound descriptions of notions and sets, and of complex discourses and knowledge pieces, in particular, of discourses with the references to the meaning of sentences or larger parts of discourse.

The approaches AMR and LCMR are rather convenient for representing structured meanings (SMs) of simple instructions: with one verb, containing no connective OR. However, the analysis shows that the expressive power of AMR and LCMR is insufficient for effectively dealing with complex instructions characterized above. The appropriate expressive mechanisms are provided only by TKR.

The part 1 of TKR is a mathematical model (Model 1) of a system of primary units of conceptual level used by an applied intelligent system. This model determines, in particular, a new class of complex formal objects called *conceptual bases.* To construct an arbitrary conceptual basis (c.b.) *B* is equivalent to defining a certain finite sequence of formal objects *Tuple(B).* The interpretation of its distinguished components *St, X, V, F, tp* is as follows [19]. *St* is a finite set of symbols called sorts and interpreted as designations of most general notions used in the considered application domains: *physical object, intelligent system, organization, distance value, price value*, etc. The countable set *V* contains the variables. The countable set *X* includes the subset *St* of sorts and contains the symbols interpreted as the designations of primary informational (or conceptual) units. The set *X* is called *the primary informational universe of the c.b. B*. The finite subset *F* of *X* contains the designations of functions.

The component *tp* of the sequence *Tuple(B)* is a function from the union of *X* and *V* into a countable set of strings *Types (B)*, it includes *St*. The elements of this set are called *types* and are interpreted as structured characteristics (labels) of the entities denoted by the elements of *X*. The mapping *tp* gives us a much more fine-grained structuring of application domains than first order logic.

Example. A c.b. *B* may satisfy the following conditions: (a) *St* includes the elements (sorts) *dyn.phys.ob* (dynamic physical object), *ints* (intelligent system), *org* (organization), *inf.ob* (informational object); (a) *X* includes the elements *M-Bulgakov, Master-and-Margaret, person, tourist-group, Suppliers, Authorship,* and

tp(*person*) =↑ *ints* * *dyn.phys.ob, tp*(*M-Bulgakov*) = *ints* * *dyn.phys.ob, tp*(*Master-and-Margaret*) = *inf.ob, tp(Authorship) = {(ints, inf.ob)},*

tp(tourist-group) = ↑ {*ints* * *dyn.phys.ob*}, *tp(Suppliers)* = {(*org*, {*org*})}.

Here the symbol ↑indicates a type of a notion; *Suppliers* is the name of the function associating an enterprise with the set of all its suppliers.

A partial order ⊢ is defined on the set of types *Types (B),* it is called the *concretization relation.* For instance, the following relationships may take place:

phys.ob ⊢ *dyn.phys.ob, phys.ob* ⊢ *ints* * *dyn.phys.ob, ints* ⊢ *ints* * *dyn.phys.ob, {phys.ob}* ⊢ *{ints* * *dyn.phys.ob}.*

The part 2 of TKR determines a mathematical model (Model 2) of a system consisting of ten partial operations on conceptual structures. The Model 2 defines, in particular, a new class of formal languages called SK-languages (standard knowledge languages). There are weighty reasons to conjecture that SK-languages are a convenient formal tool for building SRs of arbitrarily complex NL-texts (sentences and discourses) pertaining to mass spheres of professional activity (engineering, medicine, business, sport, etc.). The term "a K-representation" (KR) is used for denoting SRs of NL-texts being the expressions of SK-languages.

The expressions of SK-languages are built from primary semantic units and several service symbols by means of inductive application of some original rules P[0], P[1], …, P[10]. A set of primary semantic units and several distinguished subclasses of this set are determined by a conceptual basis (c.b.) [19]. The language corresponding to an arbitrary c.b. *B* is designated by *Ls(B).*

The mapping *tp* from the union of the primary informational universe *X(B)* and the set of variables *V(B)* into the set *Types(B)* is expanded in [19] to the mapping *tpl* from the SK-language *Ls(B)* into the set *Types(B).* For instance, the value of the mapping *tpl* for the argument *Greater(Distance(Moscow, London), Distance (Moscow, Paris))* could be the sort *prop* (meaning of proposition), it is interpreted as the type of SRs of assertions (propositions).

The rule P[0] describes an initial set of formulas from *Ls(B);* in other terms, they are called K-strings. E.g., the

unit *file1* is a K-string. The rules P[1] – P[10] jointly define a system consisting of ten partial operations on conceptual structures [19].

The rule P[1] allows us to join intensional quantifiers and designations (simple or compound) of notions, in particular, to construct the formulas *certain file1,*

*certain file1 * (Extension1, "doc"),*

*all file1 * (Extension1, "doc").*

The rule P[2] is used for constructing the expressions of the form $f(t_{1\ ,\ldots,\ t_n})$, and P[3] enables us to build the expressions of the form ($c \equiv d$). Example: *(document ≡ file1 * (Extension1, "doc"))*.

One uses the rule P[4] for building the expressions of the form $rel(t_{1\ ,\ldots,\ t_n})$, where *rel* is the name of a relation with *n* attributes (example: *Earlier (Creation-date(certain file1), #yesterday")*). The rule P[5] provides the possibility to mark a formula or its part by means of a variable. Example: *all file1 * (Extension, "doc") : S1*.

The rule P[6] allows us to join the negation connective ¬ to a formula (example: ¬*file1*). The rule P[7] governs the use of the logical connectives ∧ (and) and ∨ (or). Example: *file1 * (Extension1, ("doc" ∨ "docx"))*.

Using the rule P[8] at the last step of an inference, it is possible to construct compound designations of notions. Example: *file1 * (Extension1, ("doc" ∨ "docx"))(Location, certain desktop).*

The rule P[9] allows us to use the universal quantifier and existential quantifier (∀ и ∃) in formulas. The rule P[10] enables us to construct the SRs of finite sequences as the strings of the form $< c_{1\ ,\ldots,\ c_n} >$, where $c_{1\ ,\ldots,\ c_n}$ are the elements of a sequence.

Example 1. The algorithm SemSyntRA associates the instruction "Archive documents in folder "Project" and send to somebody@example.org" with the primary K-representation (KR) *Semrepr1* of the form

*(IsAction (#now#, archiving1 * (Object1, certain set * (Qualitative-composition, document1 * (Location, certain folder 1 * (Name1, "Project") : S1))(Result, x1), e1 ∧ IsAction (#now#, sending1 * (Object1, x1)(Email-address, "*somebody@example.org"), e2) ∧ *Immediately-after(e2, e1))*.

III. THE PRINCIPAL IDEAS OF EXECUTING K-REPRESENTATIONS OF INSTRUCTIONS

In order to execute an instruction, more exactly, a KR of an instruction, it is necessary to do two steps: (a) to form a KR including only the notions being "known" to the goal applied system; (b) to translate the obtained KR to the language of the goal system.

Let's consider the user instruction "Move the video with the name "Lecture 2017-12-15" to the reserved disc". For escaping many details, let's restrict ourselves by considering the processing of the fragment the video with the name "Lecture 2017-12-15", associated with the KR of the form *certain video1 * (Called, "Lecture 2017-12-15")*.

Suppose that we have the transformation rule

*video1 * (Called, filename) → file1*(Called, filevame)(Extension1, ("avi" ∨ "mkv" ∨ "mp4"))*,

where *filename* is a certain string such that the type *tpl("Lecture 2017-12-15")* is a concretization of the type *tpl(filename).*

Then, applying this rule to the KR *certain video1 * (Called, "Lecture 2017-12-15"),* we obtain the secondary KR of the form

certain file1(Called, filevame)(Extension1, ("avi" ∨ "mkv" ∨ "mp4"))*.

For applying this transformation rule, it is necessary to get to know that the K-string *certain video1 * (Called, "Lecture 2017-12-15")* or its certain substring corresponds to the left part of the considered transformation rule.

With this aim, a special binary relation is introduced. It is called *the relation of correspondence to a pattern* and receives the name (designation) *Match*. For the considered example, the pair (*video1 * (Called, "Lecture 2017-12-15"), video1*(Called, filevame))* belongs to *Match*.

Next section is devoted to considering the grounds for including the pairs of K-strings into a binary relation *Match*.

Similarly, it is assumed to fulfill the transformation of KR into the scripts of the goal system. The only difference will be that the right parts of the transformation rules will contain not the K-strings but the scripts of the goal system.

IV. THE GROUNDS FOR INCLUDING THE K-STRINGS INTO THE RELATION OF CORRESPONDENCE TO A PATTERN

In order to fulfill a transformation, let's formulate a rule of checking a correspondence between the left part of the transformation rule and the processed K-string. With this aim, we'll define the relation of correspondence to a pattern on the set of K-strings. The relation is defined with the help of several natural assumptions called below the grounds.

Firstly, a correspondence of an element of the primary informational universe X(B) to the pattern z may be defined by the coincidence of the meanings (synonymy). This synonymy may be determined by a certain reflexive symmetric relation. Let's introduce

Definition 1. Let *B* be an arbitrary conceptual basis (c.b.). Then the *synonymy relation coordinated with the c.b. B* is an arbitrary reflexive symmetric relation *Syn* on the primary informational universe *X(B).*

Let's determine the relation of correspondence to a pattern *Match* on the SK-language *Ls(B).* The fact that the pair *(y, z)* belongs to the relation Match will be denoted by the record $y \succ z$. We do know that the relation *Match* depends on the synonymy relation *Syn* and on the set of rules enabling us to expand *Syn* on the set *Ls(B).* That is why it would be correct to say about the relation of correspondence to a pattern with the precision to within a synonymy relation *Syn.*

Taking this into account, we formulate

Ground 1. Let *B* be an arbitrary c.b, *y* and z be the arbitrary elements of the primary informational universe *X(B).* Then the fact *(y, z)∈ Syn* implies the fact *(y, z)∈ Match*.

For instance, the pair (*folder1, catalogue1*) may belong to the synonymy relation, because the informational units *folder1* and *catalogue1* correspond to

the words "folder" and "catalogue", and these words are synonyms in the field of manipulating with files. Hence the unit *folder1* corresponds to the pattern *catalogue1*, i.e., (*folder1, catalogue1*) $\in$ *Match*, it is the same as *folder1* $\succ$ *catalogue1*.

In case the pattern is a variable, the correspondence of the K-string *y* to the pattern *z* is determined by the types of the pattern *z* and *y*. The type of the pattern *z* is to coincide or to be more general than the type of *y*. E.g., if the element of a primary informational universe *disc1* has the type $\uparrow$*inf.object * dyn.phys.object*, and a variable *var* has the type $\uparrow$*inf.object,* then the element *disc1* corresponds to the pattern *var*. We obtain in this way

Ground 2. Let *B* be an arbitrary c.b, *y* belong to *Ls(B),* and *z* be an arbitrary variable from *V(B).*Then the pair *(y, z)*$\in$ *Match* then and only then when $tpl(z) \vdash tpl(y)$, where *tpl* is a mapping from the SK-language *Ls(B)* into the set of types corresponding to *B Types(B)*.

Since a pattern may be constructed with the help of some rules P[1] - P[10], it is reasonable to see how the relation of correspondence to a pattern will be defined in each of these cases.

Let's start from considering the usage of intensional quantifiers in the K-strings (the rule P[1]). For instance, the K-string *certain folder1 * (Called, "Video")* will correspond to the pattern *certain catalogue1*, but will not correspond to the pattern *catalogue1*. Besides, a more simple K-string *certain folder1* will correspond to the pattern *certain catalogue1.*Thus, we have

Ground 3. Let *B* be an arbitrary c.b, *Syn* be a reflexive symmetric relation on X(B), *y* be a K-string of the form $intq_1c_1$, *z* be a K-string of the form $intq_2c_2$, and $(intq_1, intq_2) \in Syn$, $(c_1, c_2) \in Match$. Then *(y ,z)* $\in$ *Match*.

The K-strings with functional and relational symbols (i.e., with the names of functions and *n*-ary relations, where $n \geq 1$), built respectively according to the rules P[2] and P[4], will correspond to the patterns with similar structure.

For instance, the K-string *Size1(certain file1 * (Called, "a.txt"))* will correspond to the pattern *Size1(certain file1).* In more general cases, the functional symbols of the K-string and the pattern should be synonyms, and the arguments of a function (of a predicate) should correspond to the arguments of the pattern as the argument *certain file1 * (Called, "a.txt")* corresponds to the pattern *certain file1*.

Besides, if a variable *n* has the type *number*, the K-string *Size1(certain file1 * (Called, "a.txt"))* has the type *integer*, and the type of the sort *number* is a more general type than the sort *integer*, then the K-string *Size1(certain file1 * (Called, "a.txt")* corresponds to the pattern *n*.

We'll formulate the similar requirements to the K-strings built with the use of relational symbols and also to the K-strings of the form $(a \equiv b)$. The above said enables us to introduce three additional grounds.

Ground 4. Let *B* be an arbitrary c.b, *Syn* be a reflexive symmetric relation on *X(B*), *y* be a K-string of the form $f(a_1,\ldots,a_n)$, *z* be a K-string of the form $g(b_1,\ldots,b_n)$, where *f* and *g* be functional symbols from *F(B)*, and $(f,g) \in Syn$. Let for every *i* from 1 to *n*, the pair (a_i, b_i) belongs to *Match*. Then *(y, z)*$\in$ *Match*.

Ground 5. Let *B* be an arbitrary c.b., *y* be a K-string of the form $(c \equiv d)$, and *z* be a K-string of the form $(a \equiv b)$. Let the pairs *(c,a)* and *(d,b)* belong to *Match*. Then *(y,z)* $\in$ *Match*.

Ground 6. Let *B* be an arbitrary c.b., *Syn* be a reflexive symmetric relation on *X(B)*, *y* be a K-string of the form $r(a_1,\ldots,a_n)$, and *z* be a K-string of the form $p(b_1,\ldots,b_n)$, where *r* and *p* are relational symbols from $X(B) \backslash F(B)$. Let for every *i* from 1 to *n*, the pair (a_i, b_i) belongs to *Match*. Then *(y, z)*$\in$ *Match*.

It is obvious that the application of the rule P[5] to a K-string doesn't cause any changes in the correspondence to a pattern. Taking this into account, we formulate

Ground 7. Let *B* be an arbitrary c.b., *y* and *z* are K-strings from the SK-language *Ls(B)*, and *var* be a variable from *V(B)* such that it is possible to form the K-string *y : var* in accordance with the rule P[5]. Then it follows from (*y, z)*$\in$ *Match* that the pair *(y : var, z)* belongs to *Match*.

Consider in a simple example the case of K-strings with the left segment $\neg$ (the connective "negation"). The K-string *certain file1 * (Called, "a.txt")* corresponds to the pattern *certain file1*. It would be logical to assume that the negation of the considered unit corresponds to the negation of the pattern. That is, the K-string $\neg$ *certain file1 * (Called, "a.txt")* corresponds to the pattern $\neg$ *certain file1*.Thus, we can define

Ground 8. Let *B* be an arbitrary c.b., *y* and *z* be the K-strings from *Ls(B).* Then the fact *(y,z)*$\in$ *Match* implies the fact $(\neg y, \neg z) \in Match$.

The case of binary logical connectives $\wedge$ and $\vee$ (conjunction and disjunction) is more complex. Let's consider firstly the disjunction. i.e. the logical connective OR.

Ground 9. Let *B* be an arbitrary c.b., *y* be a K-string of the form $(y_1 \vee \ldots \vee y_n)$, and $z \in Ls(B)$. Let there be such *i* from 1 to *n* that $(y_i, z) \in Match$. Then the pair *(y,z)* belongs to the relation *Match*.

To the contrary, $y \succ (z_1 \vee \ldots \vee z_n)$ in case *y* corresponds to every substring z_i.

Ground 10. Let *B* be an arbitrary c.b., $y \in Ls(B,$ and *z* be a K-string of the form $(z_1 \vee \ldots \vee z_n)$. Let there be such *i* from 1 to *n* that $(y, z_i) \in Match$. Then *(y, z)*$\in$ *Match*.

The grounds 9 and 10 allow us also to define a correspondence to a pattern for a K-string *y* of the form $(y_1 \vee \ldots \vee y_n)$ and a pattern *z* of the form $(z_1 \vee \ldots \vee z_m)$. It is not difficult to see that $y \succ z$ in case there is such pair *(i, j)* that $y_i \succ z_j$.

Similarly to the grounds 9 and 10, it is possible to define that a K-string of the form $(y_1 \wedge \ldots \wedge y_n)$ corresponds to the pattern *z* in the case at least one of the substrings y_i. corresponds to *z*. To the contrary, a K-string y corresponds to the pattern *z* of the form $(z_1 \wedge \ldots \wedge z_n)$, if y corresponds to all substrings z_i for *i* from 1 to *n*. We use also several additional grounds, they are formulated in [13].

Example. Let's illustrate the ideas stated above in a formal way. Assume that the synonymy relation includes the pairs *(file1, document1), (folder1, catalogue1), (folder1, directory1), (disc1, carrier1), (disc1, USB)* and also the inverse (symmetric) pairs. Then the following relationships will take place:
catalogue1 $\succ$ *folder1*;

disc1≻z, if *z∈ V(B)*, *tpl(certain folder1)=inf.object * dyn.phys.object*, and *tpl(z)= inf.object* , because *inf.object ⊢ inf.object * dyn.phys.object*;
certain catalogue1≻ certain folder1;
Size1(certain catalogue1)≻ Size1(certain folder1);
*Size1(certain catalogue1 * (Called, "Docx")) ≻ Size1(certain catalogue1)*;
(certain file1 ∧ certain catalogue1)≻ certain file1;
certain catalogue1≻(certain file1 ∨ certain catalogue1).

V. Application of the Stated Formal Approach

The ideas stated above together with the model of linguistic database, the notion of a graph-like semantic-syntactic structure, and the algorithm of semantic parsing SemSyntRA [11-12] underpinned the design of file management system NLC-2 (Natural Language Commander – Version 2). This program is the next generation of NLC-1, which was developed for the studies and experiments in the field of NL-interfaces to action-based applications [9, 10]. NLC-2 processes natural language instructions in accordance with the following scheme:
User instruction ==> Primary K-representation ==> Secondary K-representation ==> BASH Script

Example. Let's look how NLC-2 processed the user instruction from the example of Section 2: "Archive documents in folder "Project" and send to "somebody@example.org"". This instruction is transformed by the algorithm SemSyntRA described in [11] into the primary K-representation *Semrepr1* described in Section 2.

Now if the knowledge base of NLC-2 contains the transformation rule *document1 → file1 * (Extention1, ("doc" ∨ "docx" ∨ "odt"))* then the system NLC-2 transforms the constructed primary K-representation of the user instruction into its secondary KR

*(IsAction (#now#, archiving1 * (Object1, certain set * (Qualitative-composition, certain file1 * (Extention1, ("doc" ∨ "docx" ∨ "odt"))(Location, certain folder 1 * (Name1, "Project") : S1))(Result, x1), e1 ∧ IsAction (#now#, sending1 * (Object1, x1)(Email-address, "somebody@example.org"), e2) ∧ Immediately-after(e2, e1)).*

Then the result shell script for Bourne-Again Shell (BASH) is as follows:

```
zip /tmp/z000129.zip "Project/*.doc"
"Project/*.docx" "Project/*.odt"; sendfile
somebody@example.com /tmp/z000129.zip
```

The final step is the execution of this script.

NLC-2 software complex contains two applications: NLC-TI – command line tool with text interface and WebNLC – tool for settings tuning and experiments with Web-interface. Both of them use the same TKRlib library. NLC-TI uses it directly. WebNLC uses TKRlib through the NLC-RESTful – a set of services for Web-applications and integration with another software.

VI. Conclusion

This paper expands new mathematical foundations of designing semantic parsers of NL instructions. The theory of K-representations underpins the stated formal approach. The principal advantages of this approach are that it is application domain independent and can be effectively used for dealing with arbitrarily complex NL instructions.

Acknowledgment

We are grateful to the anonymous referees of this paper for their efforts and precious recommendations.

References

[1] Y. Artzi, and L. Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions," in Transactions of the Association for Computational Linguistics, 2013, vol. 1, pp. 49–62. Action Editor: Jason Eisner (https://aclweb.org/anthology/Q/Q13/Q13-1005.pdf, retrieved: 2016-03-13)

[2] M. Babes-Vroman, J. MacGlashan, R. Gao, K. Winner, R. Adjogah, M. desJardins, M. Littman, and S. Muresan, "Learning to interpret natural language instructions," in Proceedings of the Second Workshop on Semantic Interpretation in an Actionable Context, pp. 1–6, Montreal, Canada, June 3-8, 2012. (https://aclweb.org/anthology/W/W12/W12-2801.pdf, retrieved: 2016-03-13)

[3] E. Bastianelli, C. Castellucci, D. Croce, and R. Basili, "Textual inference and meaning representation in human robot interaction," in Proceedings of the Joint Symposium on Semantic Processing. Textual Inference and Structures in Corpora, 2013, pp. 65-69 (https://aclweb.org/anthology/W/W13/W13-3820v2.pdf, retrieved: 2016-03-13).

[4] M. Marge and A. Rudnicky, "Comparing spoken language route instructions for robots across environment representations," in Proceedings of the SIGDIAL 2010 Conference, pp. 157–164, The University of Tokyo, September 24-25, 2010. (https://aclweb.org/anthology/W/W10/W10-4328.pdf, retrieved: 2016-03-13)

[5] L. Benotti, M. Villalba, T. Lau, and J. Cerruti, "Corpus-based interpretation of instructions in virtual environments," in Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 181–186, Jeju, Republic of Korea, 8-14 July 2012 (https://aclweb.org/anthology/P/P12/P12-2036.pdf, retrieved: 2016-03-13)

[6] D.K. Misra, K. Tao, P. Liang, and A. Saxena, "Environment-Driven Lexicon Induction for High-Level Instructions," in Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 992–1002, Beijing, China, July 26-31, 2015 (https://aclweb.org/anthology/P/P15/P15-1096.pdf, retrieved: 2016-03-13)

[7] C.S. Carlos, "Natural language programming using class sequential rules," in Proceedings of 5th International Joint Conference on Natural Language Processing, pp. 237–245, Chiang Mai, Thailand, November 8 – 13, 2011 (https://aclweb.org/anthology/I/I11/I11-1027.pdf, retrieved: 2016-03-13)

[8] T. Lei, F. Long, R. Barzilay, and M. Rinard, "From natural language specifications to program input parsers," in Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 1294–1303, Sofia, Bulgaria, August 4-9, 2013.

(https://aclweb.org/anthology/P/P13/P13-1127.pdf, retrieved: 2016-03-13)

[9] A.A. Razorenov and V.A. Fomichov, "The design of a natural language interface for file system operations on the basis of a structured meanings model," Procedia Computer Science, Elsevier, 2014, vol. 31, pp. 1005-1011; open access, URL: http://authors.elsevier.com/sd/article/S1877050914005304

[10] V.A. Fomichov and A.A. Razorenov, "Significance of the theory of K-representations for the studies on automatic semantic role labeling," ("Znachenie teorii K-predstavlenii dlya issledovanii po avtomaticheskomu vyavleniyu semanticheskikh roley"), Informatsionnye Tekhnologii, 2015, no. 6, pp. 403-411 (in Russian).

[11] A.A. Razorenov and V.A. Fomichov, "A new formal approach to semantic parsing of instructions and to file manager design," in Database and Expert Systems Applications. Proceedings of the 27th International Conference, DEXA 2016, Porto, Portugal, September 5-8, 2016. Cham: Springer, 2016, vol. 9827. Part. I, pp. 416-430.

[12] A.A. Razorenov and V.A. Fomichov, "A new approach to formalization of instructions' semantic processing based on the theory of K-representations," ("Novyi podkhod k formalizatsii semanticheskoi obrabotki predpisanii na osnove teorii K-predstavlenii"). Informatsionnye tekhnologii, 2017, vol. 23, no. 1, pp. 3-14 (in Russian).

[13] A.A. Razorenov, "The principles of applying the theory of K-representations to developing mathematical foundations of executing natural language instructions by computer systems," ("Printsypy primeneniya teorii K-predstavlenii k razrabotke matematicheskikh osnov vypolneniya estestvenno-yazykovykh predpisanii komp'yternymi sistemami"), Informatsionnye tekhnologii, 2017, vol. 23, no. 10, pp. 699-706 (in Russian).

[14] V.A. Fomichov, "A mathematical model for describing structured items of conceptual level", Informatica. An International Journal. of Computing and Informatics (Slovenia), 1996, vol. 20, no. 1, pp. 5-32.

[15] V.A. Fomichov, "Theory of restricted K-calculuses as a comprehensive framework for constructing agent communication languages," in Fomichov V.A., Zeleznikar A.P. (eds.). Special Issue on NLP and Multi-Agent Systems. Informatica. An International J. of Computing and Informatics (Slovenia), 1998, vol. 22, no. 4, pp 451-463.

[16] V.A. Fomichov, "An Ontological Mathematical Framework for Electronic Commerce and Semantically-structured Web," in Zhang Y., Fomichov V.A., Zeleznikar A.P. (eds.), Special Issue on Database, Web, and Cooperative Systems. Informatica. An International J. of Computing and Informatics (Slovenia) , 2000, vol. 24, no. 1, pp. 39-49.

[17] V.A. Fomichov, "Theory of K-calculuses as a powerful and flexible mathematical framework for building ontologies and designing natural language-processing systems," in Andreasen, T., Motro, A., Christiansen, H., Larsen, H.L. (Eds.), Flexible Query Answering Systems, 5th Intern. Conference, FQAS 2002, Proceedings, Lecture Notes in Artificial Intelligence. 2002. Berlin, Heidelberg, New York: Springer, 2002, vol. 2522, pp. 183-196.

[18] V.A. Fomichov, "A comprehensive mathematical framework for bridging a gap between two approaches to creating a Meaning-Understanding Web," Intern. Journal of Intelligent Computing and Cybernetics, 2008, vol. 1, no. 1, pp. 143-163.

[19] V.A. Fomichov, Semantics-Oriented Natural Language Processing: Mathematical Models and Algorithms, New York, Dordrecht, Heidelberg: Springer, 2010.

[20] V.A. Fomichov, "Theory of K-representations as a comprehensive formal framework for developing a Multilingual Semantic Web", Informatica, An Intern. Journal of Computing and Informatics, 2010, vol. 34 , no. 3, pp. 387-396 .

[21] V.A. Fomichov, "A broadly applicable and flexible conceptual metagrammar as a basic tool for developing a Multilingual Semantic Web," in Metais, E., Meziane, F., Saraee, M., Sugumaran, V., Vadera, S. (eds.) NLDB 2013. LNCS, vol. 7934, pp. 249-259. Heidelberg: Springer.

[22] V.A. Fomichov, "SK-Languages as a Comprehensive Formal Environment for Developing a Multilingual Semantic Web," in Decker, H., Lhotska, L., Link, S., Spies, M, Wagner, R. R. (eds.) Database and Expert Systems Applications, 25th Intern. Conference, DEXA 2014, Munich, Germany, September 1-4, 2014, Proceedings, Part I, LNCS, Vol. 8644, 394-401. Springer, Cham, Heidelberg (2014)

[23] V.A. Fomichov, "Integral Formal Semantics and the design of legal full-text databases," Cybernetica. Quarterly Review of the International Association for Cybernetics (Belgium, Namur), 1994, vol. 37, no. 2, pp. 145–177.

[24] L. Banarescu, C. Bonial, S. Cai, M. Georgescu, K. Griffitt, U. Hermjakob, K. Knight, P. Koehn, M. Palmer, and N. Schneider, "Abstract Meaning Representation for Sembanking," in Proceedings of the 7th ACL Linguistic Annotation Workshop and Interoperability with Discourse, Sofia, Bulgaria, August 8-9, 2013 (www.aclweb.org/anthology/W13-2322; retrieved 2016-03-12)

[25] V.A. Fomichov, "SK-languages as a powerful and flexible semantic formalism for the systems of cross-lingual intelligent information access," Informatica. An Intern. Journal of Computing and Informatics (Slovenia), 2017, vol. 41, no. 2, pp. 221-232.