NATIONAL RESEARCH UNIVERSITY

The Design of a Natural Language Interface for File System Operations on the Basis of a Structured Meanings Model

Alexander Razorenov, Vladimir Fomichov

Department of Innovations and Business in the Sphere of Informational Technologies, Faculty of Business Informatics,
National Research University Higher School of Economics (HSE)

Kirpichnaya str. 33, 105187 Moscow

e-mail: razorenov@mail.ru, vfomichov@hse.ru

Introduction

During last 15 – 20 years, the standard of a user interface for the interaction with applied software de-facto has been graphical user interface. Due to evolutionary increase of the software complexity, graphical user interfaces have been becoming more and more complex, and often this process encounters the restricted possibilities of reflecting graphics on a display. This situation has lead to the following unpleasant consequences:

- A user is unaware of a part of the software possibilities, because he/she doesn't find them in the endless menus, nested interface's elements, and in the opening (one after another) windows.
- The user spends a considerable time for the search or the access to the required element (in case he/she remembers the location of the required element of an interface: a button, a menu item, a checkbox, etc.).

Introduction

The analysis shows that a perspective way out of this difficult situation seems to be the design of interfaces enabling the user of a complex software system to interact with it with the help of a restricted natural language (in other terms, by means of a controlled natural language) – English, French, Russian, etc.

Here natural language (NL) interfaces (NLI) for the action-based applications (NLIABA) are considered as the complements and alternatives to traditional graphical interfaces.

State of the arts

One of the first commercial applications with NLI was Lotus HAL – a NL-interface for Lotus 1-2-3 spreadsheet application. Also there are different applications using NL-interfaces to access their functionality or to access functionality of other applications. For example:

- Siri – a personal assistant software for iPhone 4S/5;
- Sobesednik HD – a Siri's analog for Russian;
- Ubiquty – an extention to Firefox web browser;
- GNOME Do – an application for GNOME desktop environment;
- Braina Project – a software for working with computer;
- NLUI Server – a multilingual application server for NL-interface Scripts supports many languages, including English, Russian, French, German, Japan and many others.

Task statement

The mentioned applications support simple “one-action” commands: open file, create folder, copy a file to a folder, answering “how many time”. But the user commands may be much more complex.

Examples:

1. “Copy all music and text files created this year from folder A to folder B, archive them and send to e-mail address someperson@example.com”.
2. “Move documents from “Project” folder to the folder with the name “Docs” or “Documents” on backup drive if their size is less than 2 GB”

It is impossible to support such complex instructions using simple algorithms like pattern matching. A complex theory of NL-interfaces to action-based applications is required to support this kind of user commands.

The requirements to a language of semantic representations

The expressive mechanisms allowing us to describe semantic structure of the following texts:

- compound commands
- compound designations of objects
- compound designations of sets
- homogenous members of sentences

The analysis shows that now there is only one theory satisfying the mentioned requirements.

It is the theory of K-representations (knowledge representations).

This theory, developed by V.A. Fomichov, introduces, in particular, a new class of formal languages called ***SK-languages*** (standard knowledge languages).

The definition of the class of SK-languages is a part of a mathematical model developed in V.A. Fomichov monograph:

Vladimir A. Fomichov Semantics-Oriented Natural Language Processing: Mathematical Models and Algorithms. IFSR International Series on Systems Science and Engineering, Vol. 27. Springer: New York, Dordrecht, Heidelberg, London, 2010.-354 p

This mathematical model describes a system consisting of ten partial operations on conceptual structures.

Example:

Let **Operand1 = file1**

and **Operand2 = Extension(certain file1, ("doc" ∨ "docx" ∨ "odt"))**.

Then the result of applying the operation Op[8] is the formula:

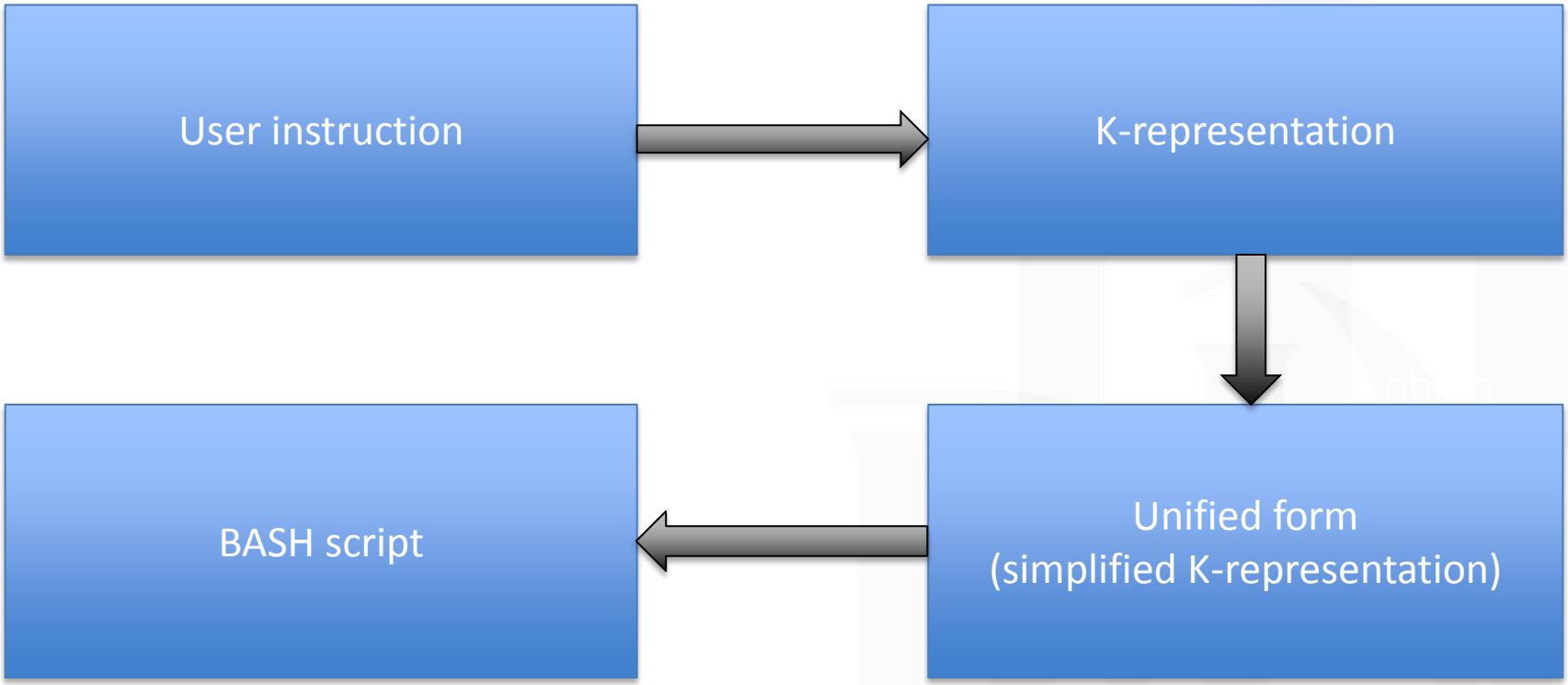
file1*(Extension, ("doc" ∨ "docx" ∨ "odt"))

This formula describes notion “file with extension doc or docx or odt”

To describe *a certain file with extension doc or docx or odt* we should add intentional quantifier *certain* by applying the operation Op[1]:

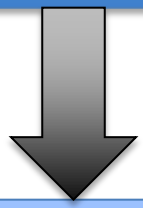
certain file1*(Extension, ("doc" ∨ "docx" ∨ "odt"))

Natural Language Commander



Natural language interface design and implementation

Move documents from “Project” folder to the folder with the name “Docs” or “Documents” on backup drive if their size is less than 2 GB



*If-then(Less(SizeOf(all Document * (Place, certain folder1 * (Name, "Project")):o1), 2/GB), Command(#Operator#, #Executor#, #now#, movement1 * (Source,o1)(Destination, certain folder1 * (Name, ("Docs" ∨ "Documents")))(Place, certain backup-drive) : o2)))*

Natural language interface design and implementation

*If-then(Less(SizeOf(all Document * (Place, certain folder1 * (Name, "Project")):o1), 2/GB), Command(#Operator#, #Executor#, #now#, movement1 * (Source,o1)(Destination, certain folder1 * (Name, ("Docs" ∨ "Documents")))(Place, certain backup-drive) : o2)))*

*(Document ≡ File * (Extention, ("doc" ∨ "docx" ∨ "odt"))),*
*(backup-drive ≡ drive1 * (Name, "F"))*

$x, x \equiv y \vdash y$

*If-then(Less(SizeOf(all file * (Extention, ("doc" ∨ "docx" ∨ "odt"))(Place, certain folder1 *(Name, "Project")): o1), 2/GB), Command(#Operator#, #Executor#, #now#, Copying1 * (Source, o1)(Destination1, certain folder1 * (Name, ("Docs" ∨ "Documents")))(Place, certain drive1 * (Name, "F") : o2)))*

Natural language interface design and implementation

*If-then(Less(SizeOf(all file * (Extention, ("doc" ∨ "docx" ∨ "odt"))(Place, certain folder1 *(Name, "Project")): o1), 2/GB), Command(#Operator#, #Executor#, #now#, Copying1 * (Source, o1)(Destination1, certain folder1 * (Name, ("Docs" ∨ "Documents")))(Place, certain drive1 * (Name, "F") : o2)))*

```
if [ $(du -cb "Project/*.doc" "Project/*.docx" "Project/*.odt" | grep total|sed -e "s/\s.*$//g") -le 1000000000 ]; then mv "Project/*.doc" "Project/*.docx" "Project/*.odt" $(ls /f/|grep -iE "^Docs$|^Documents$" | head -n1); fi
```

Conclusion

The method described in this presentation can simplify human-computer interaction.

Due to the use of SK-languages for constructing semantic representations (or text meaning representations) of the input texts, the proposed algorithm can operate with complex user commands.

That is why the elaborated method provides the possibility to create natural language interfaces to complex software.

This method allows us to develop natural language interfaces for action-based applications with support of user commands from simple to long complex instructions.

NATIONAL RESEARCH UNIVERSITY

Thank you for your attention!

20, Myasnitskaya str., Moscow, Russia, 101000
Tel.: +7 (495) 628-8829, Fax: +7 (495) 628-7931
www.hse.ru

Appendix

The basic model of the theory of K-representations provides the forms for representing structured meanings of natural language sentences and arbitrarily complex discourses pertaining to mass spheres of professional activity (technology, medicine, economics, sport, etc.).

Let us consider some important properties of SK-languages.

The collection of basic symbols of each SK-language includes two disjoint sets:

- *X* – a primary informational universe – a set of basic semantic items;
- *V* – a set of the variables.

A primary informational universe *X* includes several disjoint sets:

- *St* – the set of sorts (the designations of most general notions), its items can be used for associating the expressions of SK-languages with their semantic characteristics (they are called types);
- *R* – the set of the relational symbols.
- *F* – a subset of *R* consisting of the functional symbols

The K-representations of natural language texts are constructed from the items of the primary informational universe, the variables, and several service symbols by means of an iterative process of applying the rules of building well-formed formulas P[0], P[1], …, P[10].

The Rules determining the ten partial operations on conceptual structures

The rule P[0] provides an initial stock of formulas. For example, if the string *file* is an element of a certain primary informational universe, then *file* is a formula of correspondent SK-language.

The rule P[1] can be used to join intentional quantifiers to the concepts and to produce the formulas like *certain file, certain computer * (Manufacturer, IBM), all computer * (Manufacturer, IBM)*.

The rule P[2] can be used to construct the formulas like *f(t1,…tn)*, where f is a functional symbol, and *t1, …, tn* are the well-formed formulas. For example, *Size(certain file)* is a well-formed formula of a certain SK-language.

The rule P[3] can be used to construct the formulas with identity symbol *(a ≡ b)*. For example, *(Size(certain file)≡n12)*, *(Duration(x18))≡ 9/minute)*.

The rule P[4] can be used to construct the formulas like *r(t1, …, tn)*, where r is a relational symbol, and *t1, …, tn* are the well-formed formula. For example, *Less(Size1(certain file), 1024/byte)*.

The rule P[5] allows us to mark K-representations by some variable from the set of variables. For example, if a part of a K-representation looks like *certain file * (Extension, ".rtf") : v1*, then we can refer to the expression *certain file * (Extension, ".rtf")* in another part of a K-representation, using v1.

The rule P[6] provides the possibility to construct K-representations in the form ¬*Formula*, for example ¬ *file*.

The Rules determining the ten partial operations on conceptual structures

The rule P[7] allow us to use conjunction and disjunction in K-formulas: *(file ∨ folder)*, *(file ∧ folder)*.

The rule P[8] can be used to create compound designations of notions in the form *concept * (r[1], value[1])…(r[N], value[N])*,
where concept is an element of a primary informational universe denoting a notion, *r[1],…, r[N]* are the names of functions or relations, and the *value[1],…,value[N]* are well-formed formulas. This rule allows us to construct the *formula file * (Size,0)(Extension, "txt")* which is a K-representation of text “a file with zero size and extension “txt””.

The set of the rules described above provides the possibility to construct a K-representation of the natural language text “Empty text file is a file with zero size and extension “txt””. This K-representation is as follows:
*(file * (Property, empty) ≡ file * (Size1, 0)(Extension, “txt”))*.

The Rules determining the ten partial operations on conceptual structures

The rule P[9] allows us to use the quantifiers ∀ and ∃ like in first order logic.

The rule P[10] enables us to build the representations of ordered tuples as the formulas of the form *<a[1],…, a[N]>*, where *a[1],…, a[N]* are some well-formed formulas. This rule can be used to construct K-representations like the following:
<Place, certain backup-drive>, <Time, Midnight>, <Frequency, Everyday>.