T. V. Grishchuk, Cand. Sc. (Eng.), Ass. Prof.; N. M. Bykov, Cand. Sc. (Eng.), Prof.

DISCOURSE MODELING IN THE SYSTEMS WITH VOICE INTERFACES

Modified syntax of CF (context-free) grammars is presented. The possibility of this syntax application for obtaining the interpreted code of input commands is demonstrated.

***Key words**: CF grammar, deduction trees, speech recognition, voice interface.*

Speech is a natural form of communication between people. Therefore, man-machine interface realization in modern engineering systems based on the voice input-output of information is a prospective trend in the development of intellectual control systems.

Natural speech processing consists in the formulation and research of efficient computer mechanisms for providing natural-language communication with a computer. To the research objects belong:

- natural languages themselves;
- using natural language for both communication between people and man-computer communication.

Research task is the creation of efficient computer models of man-computer communication. Such statement of the problem distinguishes natural language processing (NLP) from the problems of traditional linguistics and other disciplines studying natural languages and allows referring it to the field of artificial intellect [1].

General and applied NLP are distinguished. The task of applied NLP consists in the development of models of using speech by a man. Applied NLP does not deal with modeling but directly with the ability of man-computer communication using natural language.

The majority of speech recognition systems (SRS) are designed for solving specific control tasks (control of devices, processes etc.) and so their grammatical level requires natural language formalization. In general case formal description of discourse with the help of context-free grammars (CF grammars) is sufficient. The most common syntax for reading CF grammars is represented by Backus- Naur forms (BNF) [2].

Deduction tree (sometimes it is referred to as a tree of syntax analysis) is considered to be the most common method for representation of a certain non-empty chain deduction. Let's consider a simple rule for deduction of one sentence: *Main (GoCommand (GoVerb ("Go"), HomeDir ("home")))*. The deduction tree for this rule is presented in fig. 1.

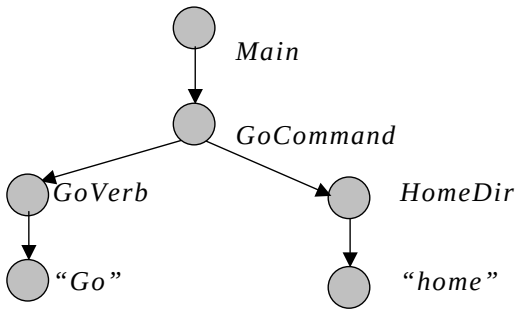

Fig. 1. Example of a deduction tree

Algorithmic and mathematical provision of the voice interface program module must solve two main problems. First, such organization of input grammar must be provided, that would make it possible to do without storing the rules themselves. Second, CF grammar syntax and mathematical model of CF grammar phrases organization must give the possibility to transform any input phrase

into the interpreted code with the ability of using structural or object-oriented approaches.

In this paper a new method of recording CF grammars is presented that provides the basis for building the grammar graph. Deduction of the grammar phrases is performed by means of going along the graph width, which enables parallel deduction of all grammar phrases in order to increase the recognition speed [3].

Vertexes of the graph are the terminals and its arcs represent grammar rules. The main disadvantage of this approach is the loss of information about the location of non-terminal symbols, which is necessary for recognition process control while going through all grammar chains of the presented graph. Non terminal symbols are represented by branching points. As one and the same terminal can play both a separate role or be a part of the non-terminal, then while generating chains on the graph a complex grammatical control is required.

This example describes the situation when a user can change the style of the text, written by him, to a “bold” or “italic” one.

Let’s write the grammar in BNF:

Grammar “Format”

< main > : make < object > < style > |

bold it;

< object > : it | sentence;

< style > : bold | italic;

This grammar is described through five terminal symbols (*make, it, sentence, bold, italic*) and 2 non-terminal symbols (*object*, *style*).

Discourse is described by the following graph:

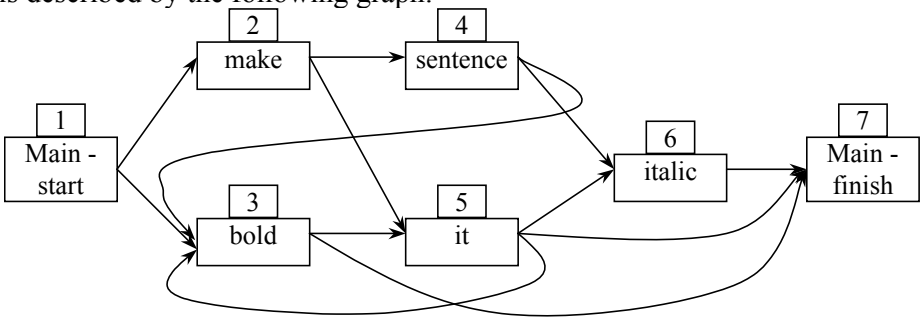

Fig. 2. Graph of the “Format” grammar

In this case it is necessary to perform grammatical control of the chain after exit from any of the terminals. Proceeding from fig. 2 for the presented graph cyclic repetition for «*bold it*» phrase is possible, which is, naturally, forbidden by the grammar.

The existing methods of grammatical deduction are also characterized by the complexity connected with different types of uncertainties, main of them being syntax uncertainty, when one and the same symbol is included into several grammar rules simultaneously. In order to eliminate such uncertainty it is necessary to introduce additional syntax control operations.

For simplification of the grammar control procedure, it is suggested to change the form of the grammar graph by introduction of additional vertexes that will represent inputs and outputs of non-terminal symbols. So each rule of the type $< S >= a < b >$ will be transformed into the form $< s\,a < b\,b > s >$.

Fig. 3 shows a modified grammar graph. From simple visual analysis of the graph it is evident that additional usage of grammar rules is no longer required.

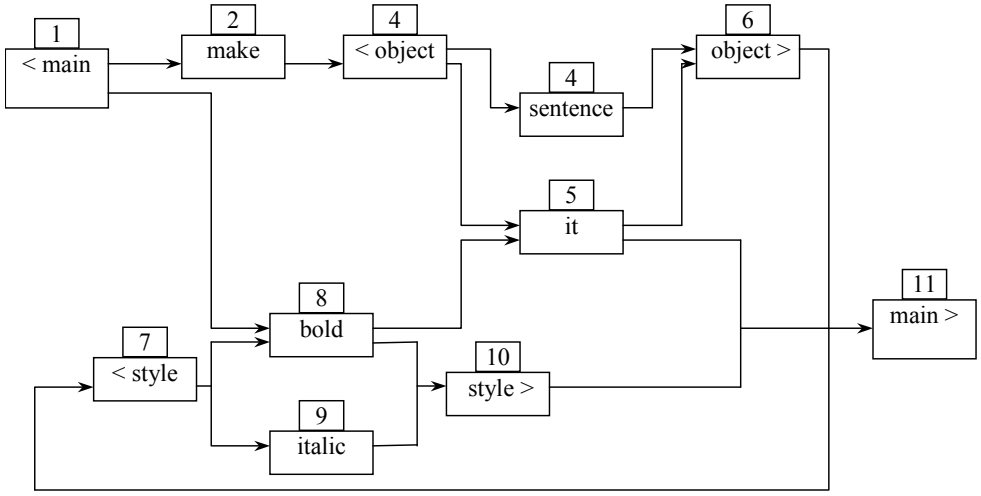

Fig. 3. The grammar graph in the modified syntax

The basic operation principle of voice interface consists in the transformation of a user's input command, given in a natural language, into a program code in the interpreter language that will be run immediately. From the previous example it is clear that classic form of recording enables simple description of the user's input command, but it requires additional means for input command transformation into output program code. In order to perform the required transformation it is necessary to obtain such deduction tree on the basis of the initial one, that would contain terminal symbols on its branches and these terminal symbols would actually be the parts of the interpreted program code.

Classical approach has the following advantages:

- input part description is separated from the output grammar building part;
- step-by-step formation of the output tree makes it possible to follow the formation process, which is helpful while editing the grammar;
- there exists a possibility to determine incorrect or undesirable commands.

The disadvantages of the approach are as follows:

- while describing transformation rules it is necessary to have a clear idea about the structure of the tree and about the influence of each step on the resulting tree;
- the complexity of changes is manifested in the following: introduction of changes at one level involves the necessity to change the rules at subsequent levels.

The second advantage of the proposed bracket syntax of grammar description consists in the ability to build the graphs of the attribute grammars on its basis. In this case elements of the output language (of the script that will finally fulfill the command) are associated with the graph vertexes. This recording method enables implementation of the structural programming principle in the process of complex grammatical structures development, e. g. at higher levels of grammar rules general structure of the given script can be described while at the lower grammar levels actual parameters can be directly obtained.

Let's consider an example of the attribute grammar. This grammar describes four phrases, each of them being translated into a certain conditional script.

```
<main(script: my_color, my_object)>:
        draw <color(my_color)> <object(my_object)>
                {script = "Draw (" + my_object + "," + my_color ");";};
        <color(color)>:
                red     {color = "red";} |
                green {color = "green";};
        <object(object)>:
                rectangle {object = "rectangle";} |
                circle       {object = "circle";}
```

In the bracket syntax this grammar has the form of:

<main draw <color color> <object object> {α_1} main>
<color red {α_2} color>
<color green {α_3} color>
<object rectangle {α_4} object>
<object circle {α_5} object>,

where:

{α_1} = {script = “Draw (” + my_object + “,” + my_color “);”;}
{α_2} = {color = “red”;}
{α_3} = {color = “green”;}
{α_4} = {object = “rectangle”;}
{α_5} = {object = “circle”;}.

Fig. 3 shows a graph of the given grammar.

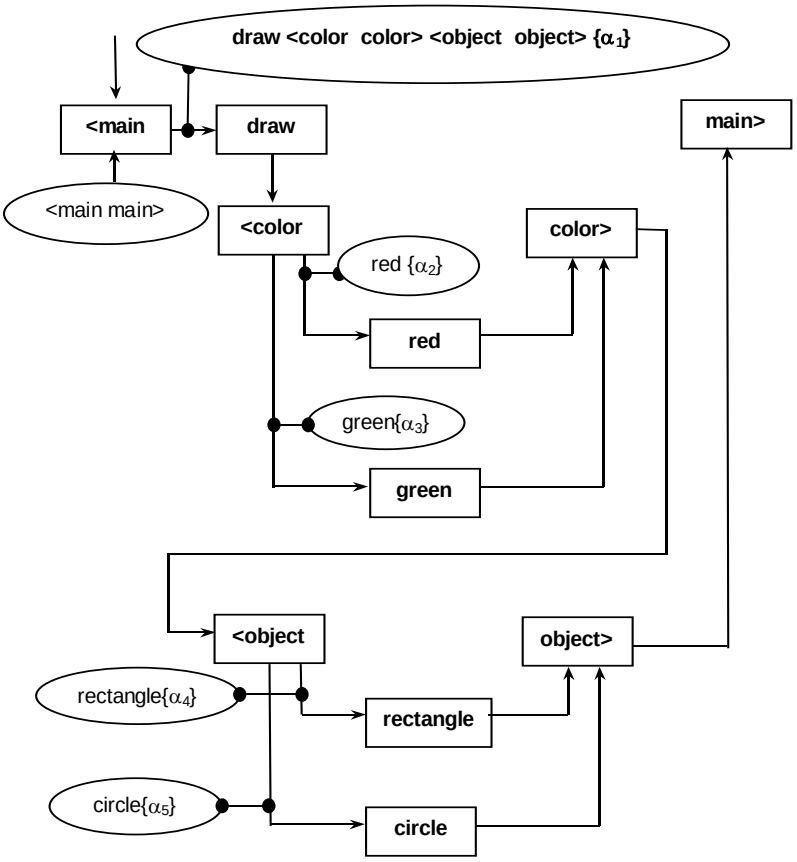

Fig. 4. The example of a graph for attribute grammar

As a result of a phrase generation on the grammatical network there is a possibility not only to receive a phrase, but also of its deduction in the bracket form. E. g. for the phrase

draw red rectangle

we receive the deduction:

<main draw <color red {α_2} color> <object rectangle {α_4} object> {α_1} main>.

Interpretation of the bracket syntax of the deduction process leads to the code generation in

the Intermediate Translation Language – ITL. For the phrase considered we receive ITL code that at the high detailization level is equivalent to the following:

```
Entry point()
{
        Declare translation;
        Main(translation);
}
Main(script)
{
        Declare my_color, my_object;
        Color(my_color);
        Object(my_object);
        script = “Draw (” + my_object + “,” + my_color “);”
}
Color(color)
{
        color = “red”;
}
Object(object)
{
oject = “rectangle”;
}
```

Interpetation of ITL code obtained in the grammatical network gives the following variant of translation: Draw (rectangle, red);.

Conclusions

Thus, the discussed method of recording CF grammars enables not only grammar representation in the form of oriented weighted graph but also parallel deduction of grammar phrases without additional grammatical control of grammatical uncertainty. Another example of this approach is the possibility to obtain the interpreted code at the stage of input command deduction using main principles of structural programming.

REFERENCES

1. Encyclopaedia of Artificial Intelligence. Entry Natural Language Understanding. – P. 660 – 677. – Режим доступу: http://sabia.tic.udc.es/encyclopediaAI/.

2. Льюис Ф. Теоретические основы проектирования компиляторов / Льюис Ф., Розенкранц Д., Стирнз Р. – М.: Мир, 1979. – 654 с.

3. Грищук Т. В. Розпізнавання природної мови на граматичних марковських мережах / Т.В. Грищук // Наукові праці Донецького національного технічного університету. Серія: “Обчислювальна техніка та автоматика”. – Донецьк: ДонНТУ, 2005. – С. 181 – 187.

Grihschuk Tatiana– Cand. Sc. (Eng.), Ass. Prof. of the Department of Computer Control Systems.

Byckov Mykola– Cand. Sc. (Eng.), Prof. of the Department of Computer Control Systems.
Vinnytsia National Technical University