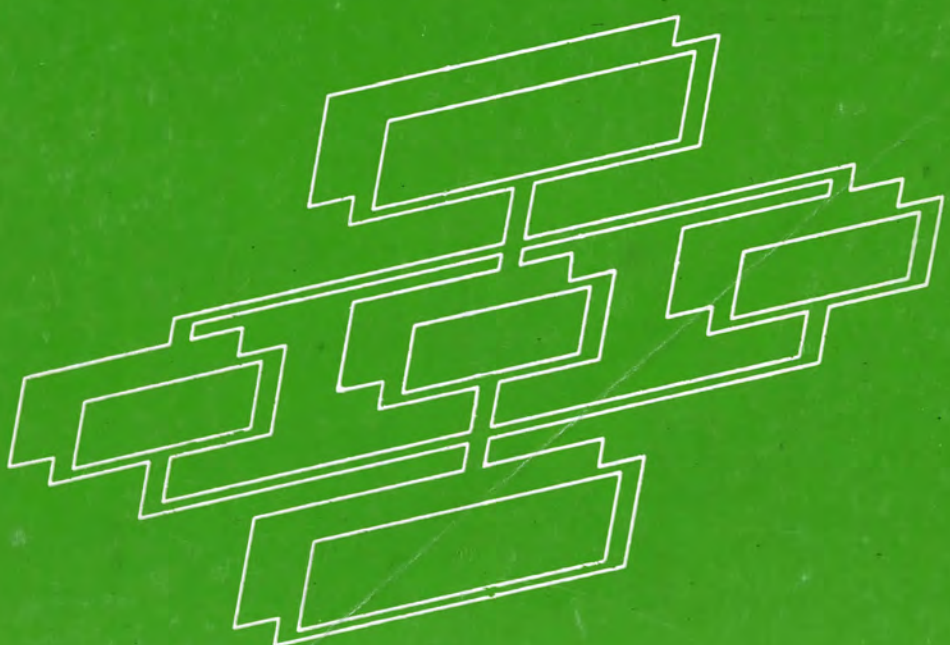
Discrete Mathematics and Applications

edited by

K. Chimev & Sl. Shtrakov



Blagoevgrad, 1993

Discrete Mathematics and Applications

edited by

K. Chimev & Sl. Shtrakov

Blagoevgrad, 1993

Contents

NATURAL LANGUAGE AND LOGIC GRAMMAR'S FORMALISMS

Radoslav D. Pavlov **Ludmila P. Dimitrova**

Institute of mathematics

Bulgarian Academy of Sciences

INTRODUCTION

When we define a language as natural as opposed to an artificial one, we have in mind that the first language is something that already exist and fulfills various functions in our dealings with other people. We use natural language to communicate with other people and we can to say all sorts of things in all sorts of situations. The formidable variety of natural languages determines its formidable complexity. The greater complexity of natural language makes the developpement of an adequate grammar and the building of an effective parser for automatic processing of a natural language considerably harder than the processing of an artificial language. Because the artificial language is something we prescribe, it has a specific purpose and is far more restrictive in the things that we can use it for. We need to pay attention to the specific features of natural language when we are trying to process it. So much

different kinds of grammar formalisms, trying to make easy the analysis of natural language, are developed. Recently, the modern trends in natural language processing are oriented towards the declarative description of natural-language grammars. Many formal grammars aim at generalizing the syntactic structure of a given language (natural or artificial) using short grammatical productions of high level. Thus, the number of grammatical rules are reduced and the analysis becomes more effective. A central part in this process have the logic grammars which use the first-order predicate logic for description of natural-language structures.

SHORT HISTORICAL REVIEW

There are some reasons why natural language is very much more difficult to process than an artificial or formal language. Natural language contains a great deal of ambiguity which is controlled in artificial or formal language by introducing of restricted rules. For example,the programming languages have a defined set of reserved words and the programmer cannot use these words for naming things. Natural language has much more complex structure than an artificial or formal language. The structure of statements in artificial or formal language is usually kept very simple, for example, in programming languages, the statements are formed by applying an operator to one or more operands, or a function to a fixed number of arguments. The same is true of more formal languages, for example, in logic: not P; A or (B and C). There appears no

simple universal way of representing the meaning of sentences in a natural language. By contrast, an artificial language is usually designed with a specific objective in mind and this can help determine how to represent the meaning. Structure and meaning are necessarily interconnected in natural languages, whereas they are often separable in artificial languages.

A. Colmerauer got the first practical realization of the idea to use logic as a language for programming. A. Colmerauer created the means for natural language analysis with PROLOG: the first logic grammars - the Metamorphosis ones. Logic grammars process logic terms into grammatical symbols and by including predicates in the rules and using logic variables, they provide a shorter description of the natural language. The Metamorphosis Grammars help axiomatizing the concatenation of terms in first order logic. It makes it possible to include the Colmerauer's q-systems properties in PROLOG I1I. So we receive a powerful instrument for a thorough syntactic and semantic processing of languages I2I. With PROLOG we can easily express the rules for the syntactic structure of the sentence and those for the semantic interpretation in natural language dialog systems. This helped the quick elaboration of the theory of logic grammars. Logic grammars are studied mainly for the purposes of automatic analysis of English. As some attempts show, however, they can be also used for processing of fragments from Spanish I3I. The use of logic grammars for adequate representation of some language phenomena in Bulgarian has not been explored up to now. The purpose of the present paper is to study the

possibilities of some kinds of logic grammars with the final aim of automatic processing of Bulgarian, taking into consideration its specific features.

SOME CHARACTERISTIC FEATURES OF BULGARIAN

The effective application of logic grammars in automatic processing of fragments of Bulgarian requirs a systematic study of these formalisms with respect to the specific nature of Bulgarian. We will outline some typical characteristic features of Bulgarian, which, to our mind , are crucial when applying logic grammars to process automatically fragments of the language. In general, for the needs of computer processing each unit (word or wordform), included in a system, using a natural language, should have the most elaborate grammatical information linguistics can give. To provide such an information is a hard and time-consuming activity. However there already exist several automatic or automated systems to make the assignment easier I4I. The accompanying grammatical information should contain all the features typical for the word base and constant for all its grammatical forms. This is due to the fact that in Bulgarian, for example, the information about gender of nouns orients the parser toward the modifying adjective in order to define the noun phrases in the sentence. The information about the transitivity of the verb - toward the object, in order to define the verb phrase, etc. So that we must know which class of words each word (lexical unit) should be assigned to. For verbs there should be

information about aspect, voice, transitivity; for nouns - about gender. Pronouns are classified according to the type: personal pronouns - person, number, gender, full or short form, sometimes case; for possessive pronouns - person, number, the possesser's gender; for the remaining pronouns - the corresponding subclass: interrogative, relative, reflexive pronouns. An example: in Bulgarian there are reflexive forms of the possessive pronouns "Moiata si", "Negovata si" and so on. This fact introduces the following semantic rule (see I10I): the short reflexive form in the dative "si" means possession when it is used as the reflexive correspondant to the short possessive form ("Dai si knigata" equivalent to "Dai knigata si"). Or when it is part of the long form of the possessive pronoun: ("Dai mu negovata si kniga" is shortened to "Dai si mu knigata"). The existance of "si" in these examples illustrates the syntactic opposition in Bulgarian: "Dai mu knigata" (the book could be his own or not) and "Dai si mu knigata" (the book is by all means his own).

SOME EXAMPLES AND APPLICATIONS OF LOGIC GRAMMARS IN BULGARIAN LANGUAGE DESCRIPTION

We will give in brief the basic terms used in formally defining of the logic grammars.

Let F is a set of functional symbols containing the binary symbol "." and "nil"(it is aceppted that a string of length zero is reduced to "nil"). Let V is a numerable set of variables. Let $\hat{H}$ is a set of terms constructed by F and V - that is a set of formulae constructed by variable or constant

(0-ary functional expression) or by expressions of type $f(t_1,t_2,\ldots,t_n)$, where f is n-ary functional symbol and t_i are terms. Let H is a set of terms into $\hat{H}$ without variables. V_T is a vocabulary called terminal and $V_T \subset H$, V_N is a vocabulary called nonterminal and $V_N \subset H$, $V_T \cap V_N = \emptyset$, $V_T \cup V_N = V$, V_S is a set, whose elements are called initial symbols, $V_S \subset V_N$. Let ---> is a binary relation between the elements of H and let $W \subset H$. Relation ---> is called rewriting relation on W^* iff for each x and $y \in H$ x ---> y implies $x,y \in W^*$. The closure of the relation ---> is notated with $\xrightarrow{*}$.

The element of the terminal vocabulary are enclosed in I I.

1. METAMORPHOSIS GRAMMARS

The Metamorphosis Grammar G, I5I, is defined as the quintuple

G = (F, V_T, V_N, V_S, --->), where ---> is rewriting relation in V^*. The language, generated by the grammar G, is the set of strings t on V_T: $L = \{ t \in V_T^* : \exists s \in V_S , s \xrightarrow{*} t \}$.

In short the rules of Metamorphosis Grammars are of the form: **sa** ---> **b**, where **s** is a nonterminal (logic) grammar symbol, **a** is a string of terminals and nonterminals, **b** is a string of terminals, nonterminals and procedure calls.

Before proceeding with an example of Metamorphosis Grammar we will deal in short with the phenomenon left extraposition. There is a left extraposition in a natural

language sentence when a subconstituent of a given constituent is missing and another constituent to the left of the incomplete one represents in a way the missing part. In this case it is useful to imagine an empty constituent called "trace" which takes over the place of the missing subconstituent. The constituent on the left that represnts the missing part, is called "marker". This marker indicates that the constituent on the right contains a "trace". For example, relative clauses have a marker which, in its simplest form, is just the relative pronoun followed by a sentence where the noun phrase has been substituted by a trace. In the sentence: **"Decata, koito Mitko srechtna, titchaha."** ("The children, that Mitko met, ran.") the subconstituent (**decata**) of the constituent (**Mitko srechtna**) is missing while the constituent on the left, the marker (**koito**), represents the missing part.

Now we will proceed with an example of MG which deals with "relativity" by shifting the trace of the object onto the verb, so that another rule could combine it with the relative marker to generate a pronoun. The MG-rules for the sentence **"Decata, koito Mitko srechtna, titchaha."** see on **EXAMPLE 1**; on Fig. 1 see the derivation graph of the noun_phrase **"Decata, koito Mitko srechtna"**.

EXAMPLE 1

(1)	sentence	--->	noun_phrase,verb_phrase.
(2)	noun_phrase	--->	noun,relative.
(3)	noun_phrase	--->	proper_name.
(4)	verb_phrase	--->	verb.
(5)	verb_phrase	--->	trans_verb,direct_object.
(6)	verb_phrase	--->	moved_dobj.,trans_verb.
(7)	relative	--->	II.
(8)	relative	--->	relative_marker,sentence.
(9)	direct_object	--->	noun_phrase.
(10)	noun	--->	**IdecataI.**

(11) proper_name ---> IMitkoI.
(12) verb ---> ItitchahaI.
(13) trans_verb. ---> IsrechtnaI.
(14) relative_marker,
noun_phrase,moved_dobj. ---> rel._pronoun,noun_phrase.
(15) relative_pronoun ---> IkoitoI.

2. DEFINITE CLAUSE GRAMMARS

We will discuss the Definite Clause Grammars (DCG), another type of logic grammars created by Pereira F.G.N. and Warren D.H.D. in 1980 (I6I). They look like context free grammars enriched with some special properties. The DCG-rules are similar to the rules of context-free grammars **nonterminal ---> <body>**, where **<body>** is the body of the production where there could be terminals, nonterminals and procedures. The possibility to add arguments to the symbols and to embed procedures in the rules determines the greater power of DCG than the context-free grammars. DCG are compiled into PROLOG clauses which define top-down or backtracking parsers in PROLOG. In the DCG-grammatical formalism each rewriting rule should expand only one nonterminal. A DCG-rule which expands a nonterminal into a string of nonterminals, looks like a standard context free rule, the difference being that when the right part of the rule has more than one nonterminal there should be an operator (similar to the comma) to combine the nonterminals into a unique term. A simple example of DCG is given by the rules:

(1) sentence ---> noun_phrase,verb_phrase.
(2) noun_phrase ---> noun.
(3) verb_phrase ---> verb.
(4) noun ---> ImagatI.
(5) noun ---> IgenataI.
(6) verb ---> ItchetaI.

This DCG defines a simple context free language, which includes the sentence **"genata tchete"** ("the woman reads").

Formally Definite Clause Grammar G is defined as follows:

$G = (V_N, V_T, V_S, P)$, where P is a set of rules of the kind:

$A \dashrightarrow b_1, b_2, \ldots, b_m$, $A \in V_N$, $m \geq 0$, $b_i \in V_N \cup V_T$, $i=1,\ldots,m$.

The language, generated by this grammar, is the set:

$$L(G) = \left\{ w \in V_T^* : \exists s \in V_S , s \overset{*}{\dashrightarrow} w \right\} , \quad V_T^* = \bigcup_{i=1}^{\infty} V_T^i .$$

It is obvious that DCG is a special case of MG , in which the rules are subject to the restriction: on the left of the rule should stand a unique nonterminal symbol, i.e. the rules are of the kind: **A ---> B**, where **A** is a nonterminal (logic-grammatical) symbol, and **B** is a string of terminals, nonterminals and procedure calls.

According to the terms of the DCG-rules we can do a simple modification I1I of the metamorphosis grammar in EXAMPLE 1, by adding a rule: **direct object ---> I I**, which allows to skip the direct object. But doing this we skip some context information, given in rule

(14) relative_marker,noun_phrase,moved_dobj.--->
rel_pronoun, noun_phrase.

Therefore we should add the rule with another argument. This argument is introduced by the symbols sentence, verb_phrase and direct_object and has accordingly the value **"I I"** when the

direct object of the sentence is not skipped and the value "elided" if it is skipped. Now the rules of the modified grammar for the sentence **"Decata, koito Mitko srechtna, titchaha."** see in **EXAMPLE 2**, on Fig. 2 see the derivation graph of the sentence **"Decata, koito Mitko srechtna, titchaha."**.

EXAMPLE 2

```
(1)  sentence              ---> sent(II).
(2)  sent(II)              ---> noun_phrase,verb_phrase(II).
(2') sent(E)               ---> noun_phrase, verb_phrase(E).
(3)  noun_phrase           ---> noun,relative.
(4)  noun_phrase           ---> proper_name.
(5)  verb_phrase(II)       ---> verb.
(5') verb_phrase(E)        ---> trans_verb,direct_object(E).
(6)  relative              ---> II.
(6') relative              ---> relative_pronoun, sent(E).
(7)  direct_object(II)     ---> noun_phrase.
(7') direct_object(E)      ---> II.
(8)  noun                  ---> IdecataI.
(9)  proper_name           ---> IMitkoI.
(10) verb                  ---> ItitchahaI.
(11) trans_verb            ---> IsrechtnaI.
(12) relative_pronoun      ---> IkoitoI.
```

3. EXTRAPOSITION GRAMMARS

Another type of logic grammars are the Extraposition Grammars (XG for short). They have been created by Pereira F. in 1981 for the implementation of a simple parsing method in the logic programming system. XG introduces the so called "skip" conception in the logic grammars theory. They are an extended version of DCG. The rules have been enriched by new properties that enable a better analysis of dynamic natural-language constructions, especially of the left-extraposition phenomenon. The left-extraposition concept, directly or indirectly, is of essential significance to a lot of formal descriptions of relative and interrogative sentences. In the cases where left-

extraposition is applied it is necessary to introduce different restrictions to avoid meaningless configurations. For the complex compound noun phrase such a restriction is: a relative pronoun that occurs out of a given noun phrase must not be connected with a trace belonging to a relative clause, subconstituent of the same noun phrase.

The XG-rule enables us to address the "skip" in its right part (the rule's) as well as to move in the same order from the right of all the constituents situated on the rule's right side. Concerning "skip" there is a restriction that requires for one to be included in the other and not to have anything in common.

Formally the Extraposition Grammar G is defined as a quintuple $G = (V_N, V_T, \hat{a}, V_S, P)$, where $\hat{a}$ is a "skip"-symbol,

$\hat{a} \in V_N \cup V_T$, P is a set of rules of the kind:

$$A, a_0, \hat{a}, a_1, \ldots, \hat{a}, a_m \text{ ---> } b_0, b_1, \ldots, b_n,$$

$A \in V_N$, $m, n \geq 0$, $a_i, b_j \in V_N \cup V_T$, $1 \leq i \leq m$, $1 \leq j \leq n$.

The language, generated by G, is the set

$$L(G) = \left\{ w \in V_T^* : \exists\, s \in V_S \,,\ s \xrightarrow{*} w \right\}, \quad V_T^* = \bigcup_{i=1}^{\infty} V_T^{\,i}.$$

We are now going to go over an informal interpretation of this definition. In order to explain the left-extraposition phenomenon we must connect two distant parts of the sentence.Neither MG, nor DCG may help to present such a connection with the help of grammatical rules. Such a connection can be achieved by the addition of supplementary information to a

lot of rules which are not grammatically tied up, and have a specific construction. Let's take a look at the following XG rules:

```
sentence                    --->  noun_phrase, verb_phrase.
noun_phrase                 --->  noun, relative.
noun_phrase                 --->  trace.
relative                    --->  II.
relative                    --->  rel_marker, sentence.
rel_marker  ...  trace      --->  rel_pronoun.
```

All of these rules are context-free. The last rule expresses the extraposition in a simple relative sentence, since it states that the relative pronoun should be analysed as a marker, followed by several unknown constituents, (shown by dots) which in turn are followed by a trace. This is illustrated by the example on the XG-rule application:

Idecata rel_marker Igenata Iluleeche trace Ipeeha --->

Idecata rel_pronoun Igenata Iluleeche Ipeeha.

The extrapositioned noun phrase is developed in a trace. Instead of being transcribed in an empty string, the trace is used as a component in relative-pronoun analysing.

According to I7I, XG should be considered as an extended view of the context-free grammars. The XG-rule differs from the DCG-rule by its property to contain a number of symbols on its left. While the DCG-rule expresses an extended version only on a single non-terminal symbol in a string, the XG-rule expresses the extended versions of a number of distant symbols in a string, all in one. For example the XG-rule permits the omission of every intermediate substring in search of the expected object, after the relative marker, and both can be

joined in a relative pronoun "**koito**", that is placed on the left side of the deleted substring.

Sometimes it is easier for the grammar rule to be considered in the direction from left to right (synthesizing), the rule can be used to extend or to rewrite the string. In the cases where the rule is used to analyse the string it's easier to consider it from right to left.

The above definition of the XG-rule application results assumes that the symbols, corresponding to the non-leading symbols on the left side of the rule, could come from arbitrarily arranged positions on the right side of the derivation. This contradicts the processing done with the context- sensitive grammars: the symbols corresponding to the context in the rule could come from positions arbitrarily arranged in the derivation, but they have to neighbour the symbol that's to be extended.

The conclusion we draw is that the XG-rules with no arguments can identify the context-sensitive languages, only when they're used for analysis.

In the XG-terms we could notate the grammar from EXAMPLE 1 by way, given by **EXAMPLE 3** (on Fig. 3 see the derivation graph of the sentence **"Decata, koito Mitko srechtna, titchaha."**).

EXAMPLE 3

(1)	sentence	--->	noun_phrase,verb_phrase.
(2)	noun_phrase	--->	noun,relative.
(3)	noun_phrase	--->	proper_name.
(4)	verb_phrase	--->	verb.
(5)	verb_phrase	--->	trans_verb,direct_object.
(6)	relative	--->	II.
(7)	relative	--->	relative_marker,sentence.
(8)	direct_object	--->	noun_phrase.

(9) relative_marker...direct_object ---> relative_pronoun.
(10) relative_pronoun ---> **IkoitoI.**
(11) noun ---> **IdecataI.**
(12) proper_name ---> **IMitkoI.**
(13) verb ---> **ItitchahaI.**
(14) trans_verb ---> **IsrechtnaI.**

The XG grammars are very useful for the economic description of the left extraposition, but are ineffective when it comes to the "skips" in the left side of the rule, "skips" should be rewritten in the respective order in the end of the rule's right side. It is also obligatory for them to go after the requirement that they should be perfectly independent of each other, or to consist completely of one another.

4. DISCONTINUOUS GRAMMARS

Discontinuous grammars (in short DG), (originally DG were called gapping grammars, but they were renamed because the term "gap" is used in linguistics with a different meaning) were devised by V. Dahl in 1981, as a generalization of Extraposition Grammars. They are basically Metamorphosis Grammars with the added flexibility that unidentified strings of constituents can be referred to (usually through a pseudo-sumbol skip(X), where X stands for the skipped substring), to be repositioned, copied, or deleted at any position. They do not need to obey, as do XG's, any nesting constraints. (XG's require that the two skips be either totally independent or that one of them be completely included in the other.) Formal description of DG, (see I8I), is the following.

DG uses a nonterminal grammatical symbol skip($\hat{a}_i$), attached to the substring, of whose structure we are not

interested at this moment, and it is between constituents, that we are interested in at such a moment. We suppose that F contains a unary symbol "skip". The discontinuous grammar G is defined as the quintuple $G = (V_T, V_N, \hat{a}, V_S, P)$, where $\hat{a} = \{skip(\hat{a}_1), skip(\hat{a}_2), \ldots, skip(\hat{a}_t)\}$, $\hat{a} \subset V_N$, is the set of "skip"-symbols, $\hat{a}i$ are variables, $\hat{a}i \in (V_N \cup V_T)^*$, $i=1,2,\ldots,t$, and P is a set of mataproductions of the form:

$$a, a_0, skip(\hat{a}_1), a_1, \ldots, skip(\hat{a}_n), a_n \text{ ---> } b_0, skip(\hat{a}i_1), b_1, \ldots, skip(\hat{a}i_m), b_m.$$

where $m,n \geq 0$, $a \in V_N$, $a_i, b_i \in (V_N \cup V_T)^*$, $skip(\hat{a}i) \in \hat{a}$ and for every j: $1 \leq i_j \leq t$. Let ===> is rewriting relation in $(V_N \cup V_T)^*$, defined as follows: $u \Longrightarrow v$ iff there exists rewriting rule

$$a_0, skip(\hat{a}_1), a_1, \ldots, skip(\hat{a}_n), a_n \text{ ---> } b_0, skip(\hat{a}i_1), b_1, \ldots, skip(\hat{a}i_m), b_m.$$

and substitution θ such that

$$u\theta = (a_0 \hat{a}_1 a_1 \ldots \hat{a}_n a_n)\theta,$$
$$v = (b_0 \hat{a}i_1 b_1 \ldots \hat{a}i_m b_m)\theta, \text{ for some } \hat{a}i \in (V_N \cup V_T)^*.$$

The closure of relation ===> is notated by $\overset{*}{\Longrightarrow}$.

The language, generated by this grammar G, is the set

$$L(G) = \{w \in V_T^* : \exists s \in V_S, s \overset{*}{\Longrightarrow} w\}.$$

A "skip" can be imagined as an unknown string of constituents to be repositioned (or copied, or deleted) by application of the discontinuous (DG)-rule.

The simple grammar from **EXAMPLE 4** handles sentence coordination and reconstitutes the meaning representation of an elided object (on Fig. 4 see the derivation graph of the sentence

"Maria vidia i Ivan tchu vlaka." "Maria saw and Ivan heard the train.").

EXAMPLE 4

```
(1) sentence                          ---> sentence,coconj,sentence.
(2) sentence                          ---> name,verb_phrase.
(3) verb_phrase                       ---> verb,object.
(4)   verb_phrase                     --->
                                     verb_phrase,coconj,verb_phrase.
(5) object                            ---> noun.
(6) object.coconj,skip(â1),object     --->
                                          coconj,skip(â1),object.
(7) noun                              ---> IvlakaI.
(8) name                              ---> IIvanI.
(9) name                              ---> IMariaI.
(10) verb                             ---> IvidiaI.
(11) verb                             ---> ItchuI.
(12) coconj                           ---> IiI.
```

The rule 6 for "object" elided an expected "object", followed by a skip (and reconstructs its internal representation). The skip between "coconj" and the "object" is recopied unanalyzed.

The augmented use of DG includes: left extraposition with more "skips", right extraposition with more "skips", interaction between different "skip"-rules, free word order; and ensures a hight level of grammatical formulations.

CONCLUSION

When logic grammars are used in systems for automatic processing, these grammars can define linguistic structures which do not appear in the considered natural language. In such cases, as well as when the grammar must be extended in order to cover a wider fragment of the natural language, the rules must be

restricted (corrected) in the right direction, so that later no corrections of the extended grammar will be needed I9I.

In this paper, we have briefly considered four kind of logic grammars and some aspects of the possibilities of their application for the purposes of automatic processing of Bulgarian, taking into consideration its specific characteristics. The examples, shown here, have been chosen simple in order to provide clear illustration for the possibilities of logic grammars for describing Bulgarian language grammar although partially.

REFERENCES

1. Colmerauer A. les systemes-q ou un formalism pour analyser et synthetiseur des phrases sur ordinateur. Departement d'Informatique, Universite de Montreal. 1970.

2. Colmerauer A. Un systeme de communication homme-machine en francais. Universite d'Aix-Marseille. 1973.

3. Abramson H., Dahl V. Logic Grammars. Springer Verlag. 1989.

4.Dimitrova L., Isusova N. A System for Automatic Retrieval of Linguistic Information. MTA SZTAKY Tanulmanyok. 182. Budapest. 1986.

5. Colmerauer A. Matamorphosis grammars. in L.Bolc (ed.) Lecture Notes in Computer Science, vol. 63. 1978.

6. Pereira F.C.N., Warren D.H.D. Definite Clause grammars for language analysis. Artificial Intelligence. vol. 13. 1980.

7. Pereira F.C.N. Logic for Natural Language Analysis. SRI International, Technical Note 275. 1983.

8. Dahl V. Hiding Complexity from the Casual Writer of Parsers. in V.Dahl, P.Saint-Dizier (ed.) Natural Language understanding and logic programming. North-Holland. 1985.

9. Stabler E.P. Restricting Logic Grammars. Computational Linguistics. vol. 13, number 1-2, 1987.

10. Й.Пенчев. Строеж на българското изречение. София. 1984.

The MG-derivation graph of the noun-phrase:

"Decata, koito Mitko srechtna"

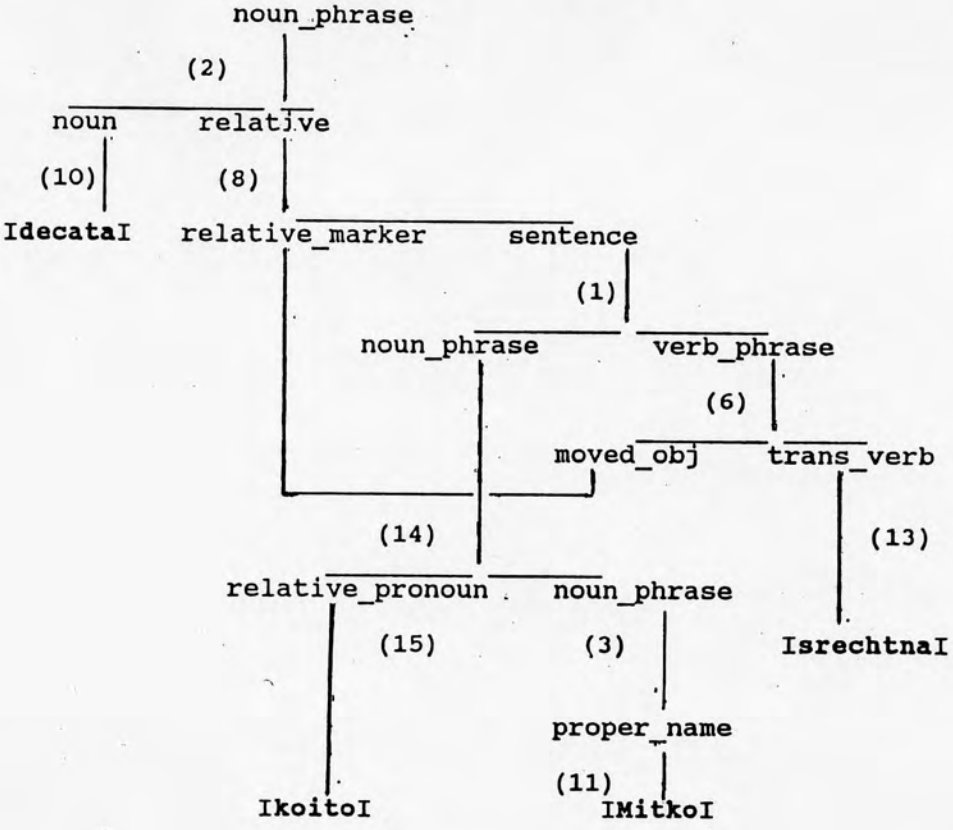


Fig. 1

The DCG-derivation graph of the sentence:
"Decata, koito Mitko srechtna, titchaha."

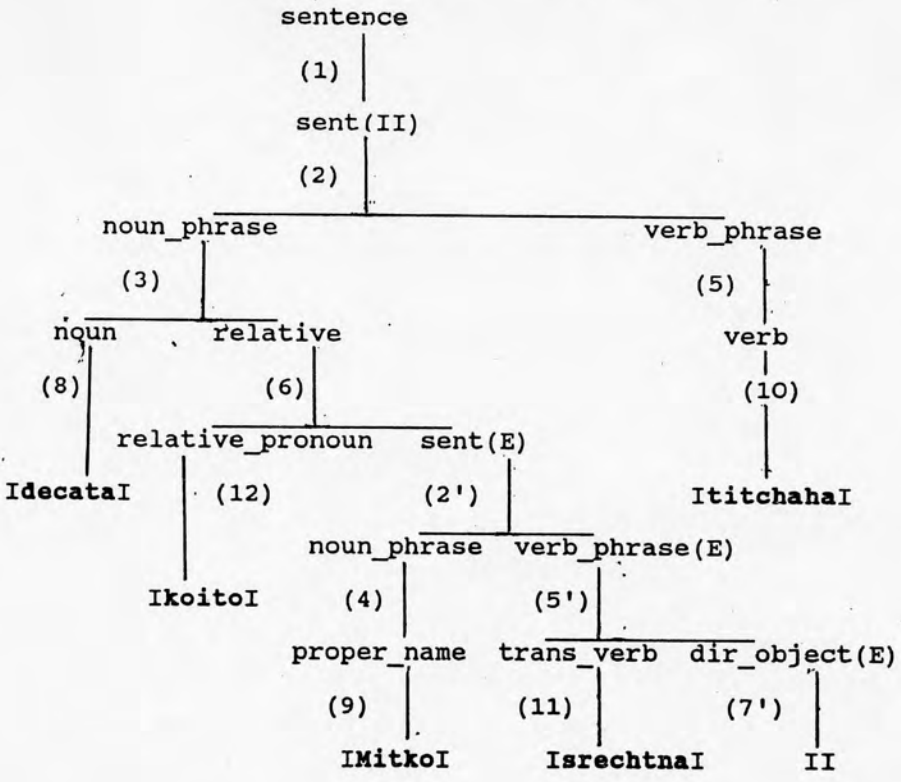


Fig. 2

The XG-derivation graph of the sentence:
"Decata, koito Mitko srechtna, titchaha."

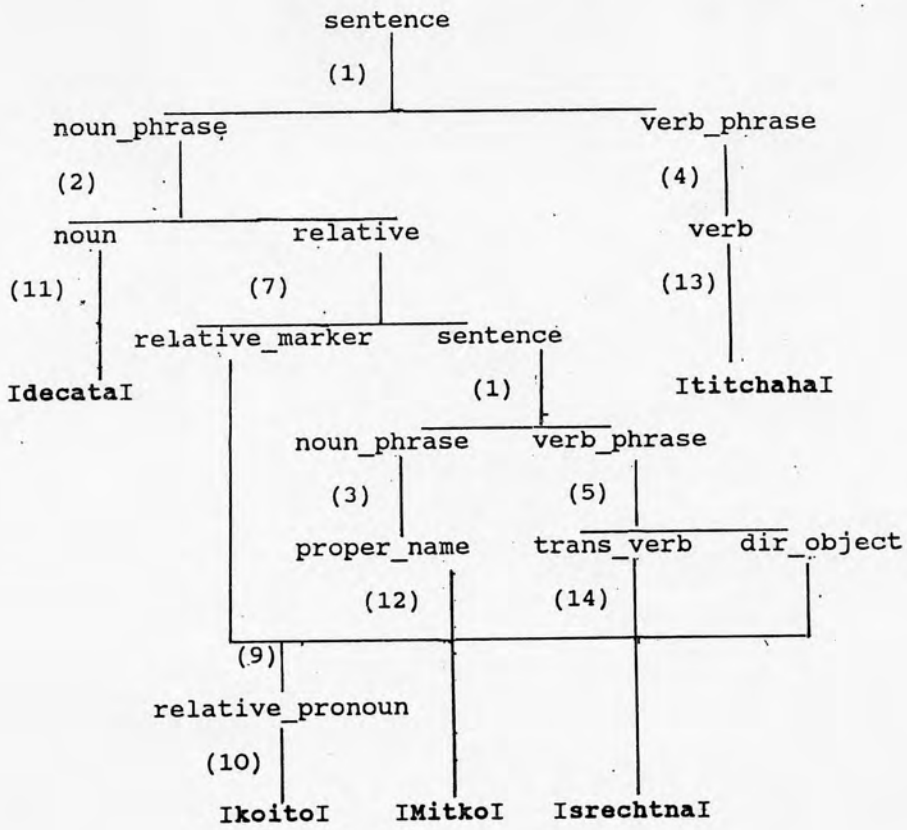


Fig. 3

The DG-derivation graph of the sentence:
"Maria vidia i Ivan tchu vlaka.".

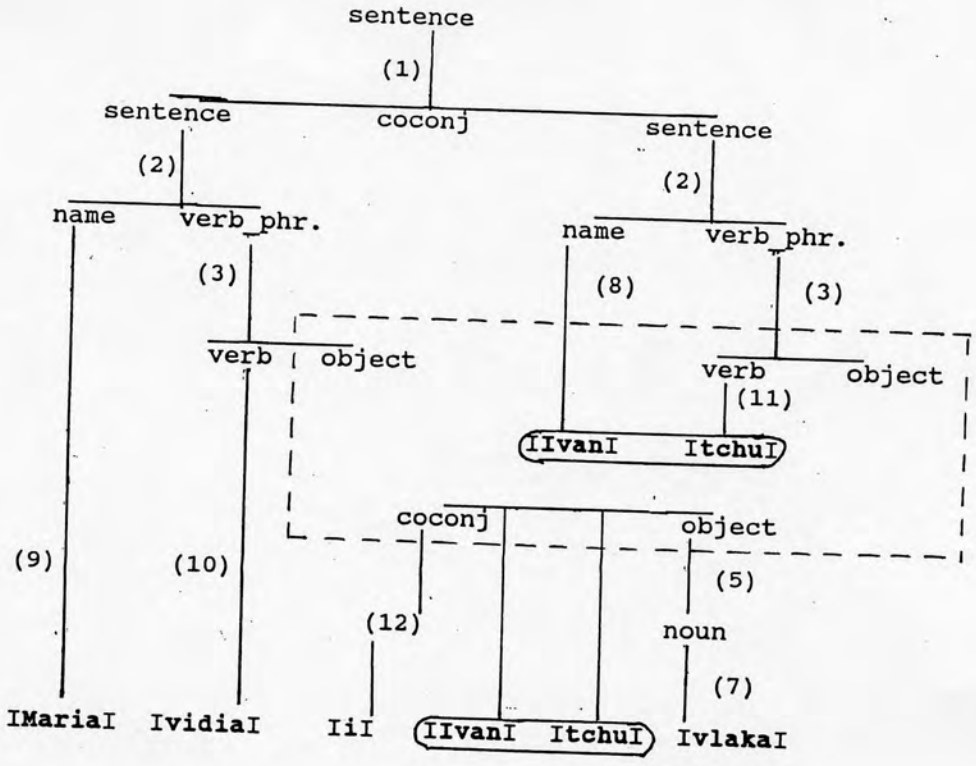


Fig.4