KYBERNETIKA — VOLUME 11 (1975), NUMBER 1

On Certain Aspects of Generative Grammar Computer Testing

ANTONÍN ŘÍHA, SVATAVA MACHOVÁ

The present paper deals with the description of computer testing of the functional generative description (proposed by P. Sgall). Illustrations are given of the system of programmes for testing the generative component of the grammar, and some aspects of notation necessitated by the specific requirements of computer testing are pointed out.

1.

The problem whether a particular generative grammar is working in such a way as it is assumed to, is unsolvable by a linguist not aided by computer. Therefore, programmes started to be written, so as to enable the computer to test particular variants of generative description (cf. [1; 2; 3]).

In the Computing Centre of Charles University in Prague tests are being worked out for functional generative description of the Czech language. Some preliminary tests were carried out earlier on smaler types of computers (LGP 30 – 1965; Minsk 22 – 1969). The aim of the present paper is to describe the preparation of the programmes for testing the generative component (GC) of a certain variant of functional generative description (cf. [4; 5]), enumerating semantic representations (SR's) of sentences.

Before demonstrating the way in which the testing is carried out we shall give a brief outline of the basic properties of the GC of the functional generative description.

2.

The GC is a context-free phrase structure grammar, i.e. a grammar of the type 2 in Chomsky's classification of grammars. Together with the theory of immediate constituents the dependency syntax finds its application in it. In the GC there are several types of context-free rules: modifying, substitutional and selectional ones. They are shown illustratively in Fig. 1, where U, V, W are auxiliary non-terminal symbols, u is a terminal symbol, r is a functor of two arguments – terminal symbol indicating which of the two non-terminal symbols on the right-hand side of the rule is the governing one and which is the dependent one as well as the type of dependency. For the sake of simplicity each of the terminal and non-terminal symbols is represented by a single letter.

REWRITING RULES	modifying rules	$U \rightarrow (VrW)$
	substitutional rules	$U \rightarrow V$
	selectional rules	$U \rightarrow u$

Fig. 1.

A substitution of some units for others is often possibic in Czech in certain contexts only. The GC meets this fact generally by introducing new, more refined categories specified by indices.

In the written form of the grammar, non-terminal symbols of the grammar are ordered $(n + 2)$-tuples $X, X_0, X_1, \ldots, X_n$ where X is the so-called name-symbol, i.e. a name shared by a certain class of non-terminal symbols, and $X_0, X_1, \ldots, X_n$ are indices specifying individual non-terminal symbols of that class. Terminal symbols are characterized by a similar structure. From a linguistic point of view the name-symbols in terminal symbols correspond to the so-called lexemes, grammatemes, and functors. (One name-symbol corresponds to the left bracket and one to the right one.) For the time being the total number of lexemes in the variant tested is 275. Thus this experiment ranks among those experiments operating with a "lexicon" of a small extent which is typical of most computer experiments with generative grammar so far.

The number of indices actually used varies with the individual name-symbols the brackets and some other name-symbols having no indices at all. The maximum number of indices attached to one non-terminal name-symbol is 15; the maximum number of indices attached to one terminal name-symbol is 30; the maximum number of values of each individual index is 94, and the average number of values per one index is 8.

3.

The form of grammar was maintained in a shape close to the original one, with which the linguists are used to work, and which makes a good orientation in the grammar possible. Some modifications, however, were carried out. The values of the indices were coded with natural numbers and some designations were introduced, such as the so-called references, which make it possible to do only one registering of the lists of index values and even of the whole non-terminal symbols that occur more than once, and to refer to these values in other cases. These lists, symbols and values are usually referred to by means of references only in the frame of a single rule-scheme so that these references do not cause any serious slowing down of the work of the programme and at the same time they save the storage space.

There are some quantitative characteristics of the variant tested here (the given values are mere estimates based on partial calculations, exact values will be reached on the computer in the course of transducing the grammar into a form suitable for the work of the programme):

a) the number of non-terminal name-symbols	62
b) the average number of indices with one non-terminal name-symbol	5
c) the average number of possible values of one index with one non-terminal name-symbol	8
d) the number of terminal name-symbols	304
e) the number of "proper" name-symbols corresponding to lexemes	275
f) the average number of indices with one terminal name-symbol	13
g) the average number of possible values of one index with one terminal name-symbol	7
h) the average number of schemes of left-hand sides for one non-terminal name-symbol	7
j) the total number of schemes of left-hand sides	460
k) the average number of schemes of right-hand sides for one left-hand side	2
m) the total number of schemes of right-hand sides	1010

The leftmost-derivation method is used as well as a random choice of alternatives. It was, however, necessary to avoid a repeated choice of some e.g. recursive rules, which would lead either to an excessive prolongation of the string generated, possibly without any transition to terminal symbols, or to a situation where the number of some types of SR's in the generated sample would be far removed from their actual frequency in the language. Therefore, the alternatives on the right-hand sides are not picked out at random only, but with a certain prescribed probability. In consequence of the use of rule-schemes it is not possible to prescribe the probabilities only to the schemes of the right-hand sides as wholes, but it is necessary to prescribe also probabilities for various values of indices used in these schemes. The prescribed probabilities also enable us to control the derivation of the SR's so as to make the generated sample contain, first of all, some strings of a particular type chosen a priori, which we intend to examine more closely. Thus it will be possible to change the set of the generated strings by means of a change of the probabilities prescribed. Here is an example of the form of the rule-scheme in the written form of the grammar (the modifying rule):

```
VERBUM  0 = 18  7 = 1 (50), 2  8 = 0,1  $ 9 = 13  →  40  (NP  0 = 9 (20), 10 (30), 11
                                                            2 = 1  3 = L8  RD  LS)
```

VERBUM, NP	are non-terminal name-symbols
RD	is a functor (terminal name-symbol)
7 = 1 (50), 2	in 50% of cases the non-teminal symbol "*VERBUM* ... *7 = 1* ..." will be rewritten according to the above rule, in the rest of cases it is necessary to use the next suitable rule
8 = 0, 1	the rule can be used when the value of index 8 with the given non-teminal symbol is 0 or 1
$ *9 = 13*	the rule can be used for the non-terminal name-symbol *VERBUM* when index 9 either has the value 13 or is not used
40	the prescribed probability for the choice of this alternative (other alternatives are not quoted here for this example)
0 = 9 (20), 10 (30), 11	in 20% of cases the value 9 is to be chosen; in 30% of the rest of cases the value 10 is to be chosen, otherwise the value 11
3 = L8	the value of index 3 with the non-terminal symbol *NP* will equal the value of index 8 of the rewritten non-terminal symbol
LS	another type of reference; in this place the whole left-hand side should be repeated (i.e. *VERBUM* with all its indices).

4.

The most essential work involved in the transduction of the grammar into a shape which is convenient for the work of the computer was left to the GRAMMAR TRANSDUCER programme, which also performes the input check of the representation. The programmes are being prepared for the computer of the type IBM 370 and will be written in PL/1 programming language.

The whole grammar will be written on a magnetic disk by means of the GRAMMAR TRANSDUCER programme as one file. This file will contain records of variable lengths. Each record will contain information about all left-hand sides with the same name-symbols and about the corresponding right-hand sides.

The main programme – DERIVATION OF SEMANTIC REPRESENTATIONS – and the generated string will be stored in the internal storage of the computer. The programme reads a corresponding record of the file on the disk storage, using the code of the non-terminal name-symbol as a key. The record read is processed directly in the buffer with the use of based variables; first, an appropriate left-hand side of the rule and then a corresponding right-hand side of it are found and then the substitution is carried out. On the output of the programme there will be two output files containing generated semantic representations. One of them will be recorded on the magnetic tape and will serve as an input for the next (transductive) component of the generative system, the other will be a file to be printed and to serve for checking up the results.

Possible changes and corrections of the rules of the grammar will be carried out by means of the programme called GRAMMAR MODIFIER, which will carry out the changes in the file stored on the magnetic disk according to the data on punched cards. A system of programmes for testing the whole generative grammar is illustrated by means of a flow-chart diagram (Fig. 2).

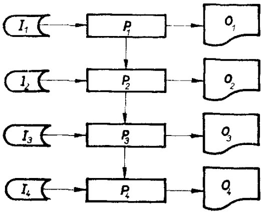

Fig. 2.

Input files:

I_1 Rules of Grammar
I_2 Rules of Transduction
I_3 Rules of Transduction
I_4 Rules of Transduction

Processes:

P_1 Derivation of Semantic Representations
P_2 Transduction to the Phenogrammatical Level
P_3 Transduction to the Morphemic Level
P_4 Transduction to the Graphemic Level

Output files:

O_1 Semantic Representations of Sentences
O_2 Representations of Sentences on the Phenogrammatical Level
O_3 Representations of Sentences on the Morphemic Level
O_4 Sentences in Graphemic Form

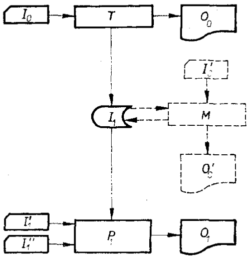

Fig. 3.

Input files:

I_0 Vocabularies and Rules of Grammar
I_0' Corrections and/or Modifications
I_1' Initial Pseudorandom Number
I_1'' Preprocessed Initial String for the Derivation

Processes:

T Grammar Transducer
M Grammar Modifier
P_1 Derivation of Semantic Representations

Output files:

O_0 List of the Rules of Grammar
O_0' List of the Modified Rules of Grammar
O_1 Semantic Representations of Sentences

Output/Update/Input file:

I_1 Vocabularies and Rules of Grammar

The diagram of programmes corresponding to the GC is given in Fig. 3. The other data will be prepared in an analogous way.

5.

The algorithm for generating semantic representations as well as the algorithms for the transduction to the other levels are taken from the theoretical description of the system. Bulk of work at writing each of the programmes is connected with the determination of the structure of the data-files and data-sets distribution in the internal or external storage. It is very important and even decisive from the point of the promptness of the operations of the programme to ensure the number of transmissions between the internal and external storage to be minimum and the search in the files to be as effective as possible.

Our method of computer testing of the generative grammar, i.e. making the computer generate representations of sentences, is similar to that developed at Michigan (see [3]), at Stuttgart (see [1]) as well as other research centres. The main difference between Friedman's test and that carried out here lies in the fact that two different types of generative grammars are under test, viz. a transformational generative grammar in the former and a functional generative grammar working with a semantic base in the latter.

As far as the questions of notation applied and the component algorithms are concerned, a number of identical problems is faced in the testing of either of the above types of grammars. In both a formalized notation of representation of the rules of grammar and other data has been worked out. Unlike the system of J. Friedman, a grammar without any ordering of rules is made use of in the test for the functional generative grammar.

In each of the two tests a different kind of control of the derivation process is applied. J. Friedman utilizes so-called skeletons, constraints and node variables — i.e. means making it possible to determine to a large extent the structure of the generated sentence in advance. E.g. the skeletons make it possible not to start from the initial symbol in the process of generation, but to start from some pre-established string (which will form a part of the sentence generated); this process, among others, may be used for testing some group of rules chosen. Apart from the above means a random choice is made from among the possible variants of rules. In testing the functional generative grammar the derivation is controlled by a system of probabilities prescribed for the choice of individual variants. Like in the tests of J. Friedman, in this kind of testing, as well, it is not necessary to start the derivation process from the initial symbol, and it is possible to start also from some other pre-established string. The ordering of rules with Friedman makes it possible to proceed in generating "from the rules to the string", i.e. to see whether the left-hand side of the rule in question occurs within the string already generated: only when no such occurrence is found can one proceed to the next rule where a similar procedure is applied.

In testing the functional generative grammar a reverse direction of the procedure is applied, "from the string to the rules": for the occurrence of a complex non-terminal symbol in question a corresponding left-hand side is searched for.

The testing of transformations involves, to some extent, similar problems as that of transductive components of functional generative grammar. However, a more detailed comparison of testing two different types of generative grammars (transformational and functional) can only be attempted when programmes have been worked out for testing the whole functional generative grammar of Czech, i.e. not only programmes for testing its GC but also those for testing its transductive components.

6.

The main contribution of the computer testing of a generative grammar is usually seen in the fact that the linguist acquires knowledge on the interaction of the rules of grammar [3]. Moreover, the preparation of programmes and the data for the computer calls for more accurate formulations and solutions of some questions, which, otherwise, would be neglected as less important.

We also consider it useful that the results of computer testing of any particular generative grammar can help even a linguist who is not familiar in detail with the theory in question: he can promptly verify his own opinion or that of somebody else on linguistic properties of the sequence generated, and gain useful information for his own futher research. For, any linguist, even when supplied with a thorough description of the properties of the generative grammar, can gain a rough idea on the strings to be generated, but it is beyond his capacities to conceive a detailed picture of the system. It is through a study of sets of strings generated by computer that he is able to make his conception more accurate and to draw conclusions as to the extent to which the strings meet his objectives or his linguistic intuitions. Then he can decide aptly whether the generative grammar concerned will be applicable in his own work.

(Received November 30, 1973.)

REFERENCES

[1] I. Batori: Working with the Interactive Version of J. Friedman's Grammar Tester. Intern. Conf. on Computational Linguistics, Pisa 27. 8.—1. 9. 1973.

[2] S. Boisvert, A. Dugas, D. Bélanger: OBLING: A Tester for Transformational Grammars. Intern. Conf. on Computational Linguistics, Pisa 27. 8.—1. 9. 1973.

[3] J. Friedman et al.: A Computer Model of Transformational Grammar. American Elsevier, New York 1971.

[4] P. Sgall: Generativní popis jazyka a česká deklinace. Academia, Praha 1967.

[5] P. Sgall, L. Nebeský, A. Goralčíková, E. Hajičová: A Functional Approach to Syntax: A New Type of Generative description of Language. American Elsevier, New York 1969.

Antonín Říha, prom. mat., PhDr. Svatava Machová, CSc.. Centrum numerické matematiky matematicko-fyzikální fakulty UK (Faculty of Mathematics and Physics, Charles University, Centre of Numerical Mathematics). Malostranské nám. 25, 118 00 Praha 1, Czechoslovakia.