OBLING: A TESTER FOR TRANSFORMATIONAL GRAMMARS

1. INTRODUCTION

Transformational grammars have developed with recent research in linguistics. They appear to be a powerful and explicit device for characterizing the description of sentences; they also meet conditions of adequacy that can be applied to check that a sentence, or a set of rules, is well-formed.

A transformational grammar tester is part of a strategy for the selection of a well-formed grammar matching the data base. To put it more explicitly, a tester of this sort should provide the linguist a class of the possible grammars which concerns precisely the linguistic theory. These grammars have the form given by Chomsky in Aspects (see bibliography).

2. GENERAL DESCRIPTION OF THE SYSTEM

OBLING is a program for dealing with the structures of the French language: it performs the verification of phrase structure rules, the derivation of sentences according to the transformational component and the graphic illustration of the intermediate or final structures.

In the program, LING is the routine that controls all the subroutines and the matching of the input structures with those allowed by the phrase structure rules. If the matching is impossible, a comment is

Acknowledgments. This work was supported in part by Canada Council of Arts grants #69-0404 and #71-0819.

We are also indebted to the staff of the Computer Center of the Université du Québec à Montréal for providing computing facilities, and giving this project high priority.

David Sankoff, of the Université de Montréal, is also responsible for the first version of the tree editing program. Finally, Josselyne Gérard helped debugging linguistic rules.
Fig. 1. Arbre syntaqtique

Fig. 1. Tree for an input sentence
The system OBLING is divided into four parts: a main program LING, the service subroutines, the phrase structure grammar tester and the transformational grammar testers. LING and the service subroutines are stored in the central memory while the two grammars testers operate on disks.

The main program invokes the various linguistic rules and controls the application of these rules to the data base structure(s) or the derived structure(s). The service subroutines are called by the routines concerning the application of the transformational rules and work in parallel with LING during the processing.

Fig. 2. The OBLING system

4. SPECIFIC CAPABILITIES OF THE PROGRAM LING

The program LING will first initialize the working areas. Then, it loads and operates the program VERIFICATEUR which, after the reading and the verification of the input data, returns control to LING.
LING will then load and execute, using an overlay technique, the small control programs CYCLI Q1, CYCLI Q2, ..., CYCLI Q·. Each of these handles, in conjunction with LING, the mapping on the input structure of a fixed number of transformation rules. In the current version of the program, CYCLI Q1 deals with the linguistic transformational rules $T_1$ to $T_s$ included, CYCLI Q2 the rules $T_s$ to $T_{10}$ included, etc. The total number of these control programs CYCLI Q depends on the memory space allowed; processing is most efficient if the number of these control programs is as small as possible.

5. INFORMATION PATTERN BETWEEN LING AND VERIFICATEUR

When VERIFICATEUR (the phrase structure grammar tester) is in memory, the structure to be analysed is read from the standard input unit (punched cards) and is processed by the subroutine CKARBRE to

Fig. 3. The VERIFICATEUR program (see figure 1 for updated tree and structure)
be validated. This subroutine first checks if the phrase structure is consistent, then calls up CRK which tests the names of the constituents describing the structure; finally, it compares this structure with those allowed by the phrase structure rules.

When errors are discovered during the processing, various sorts of comments are printed and followed if possible by a partial or full tree of the sentence. When updating is done, the tree is printed and the program VERIFACATEUR passes control to LING. The following illustrations concern first, the program VERIFACATEUR and second, an example of an updated tree and structure.

6. INFORMATION PATTERN BETWEEN LING AND THE TRANSFORMATIONAL GRAMMAR TESTERS

Each time LING receives the control from VERIFACATEUR, that is, when no further errors have been detected, it loops in order to call successively the monitors CYCLI Q1, ..., CYCLI Q9 which contain up 45 different rules; we suppose that we are working now with a specific version of a grammar.

The first of these monitors has the following structure.

| Transformational rule # 1 |
|---------------------------|
| Transformational rule # 2 |
| Transformational rule # 3 |
| Transformational rule # 4 |
| Transformational rule # 5 |

Fig. 4. The CYCLI Q1 program

When CYCLI Q1 gets control, it is bound to the application of $T_1$, ..., $T_5$ which correspond to the first five transformational rules; then control is switched to LING which calls CYCLI Q2. The programs CYCLI Qn process cyclic rules and the output structure is the input structure for the following rule. When all the cyclic rules have been applied to the input structure, LING starts over again at CYCLI Q1. If no modifications
to the already processed structures occur, or if new errors are discovered, control returns to LING.

After all the cyclic rules have been applied, the post-cyclic rules are processed in a similar manner: $CYCLI\ QA$ comprises the first five post-cyclic rules $CYCLI\ QB$, the five following, and so on.

This chart illustrates the general interaction between the programs for the processing of cyclic or post-cyclic rules.

![Flowchart](chart.png)

**Fig. 5. Flow of Control between control programs under the direction of LING**

7. SERVICE SUBROUTINES

They are implemented within the main monitor LING. All but a few of these subroutines are called during the execution of the routines corresponding to the 88 rules, that is during the phrase structure analysis or the mapping of $n$ structures.

A short description of the main subroutines follows: $ARBRB$ (tree). This subroutine is responsible for printing the tree. At the input, we find a vector $D$ of $ND$ elements which represents the tree. The horizontal distance for printing is calculated along with the total number and the relative position of these nodes; the vertical one is fixed.
OBLING: A TESTER FOR TRANSFORMATIONAL GRAMMARS

For example,

This subroutine is needed when nodes are erased; another subroutine, NEWTREE will erase the nodes. In the example below, OTE sets $D(6), D(7), D(13)$ to zero, and NEWTREE erases nodes numbered 6, 7 and 13. If node 12 was also erased, OTE and NEWTREE would have erased node 28 automatically. The same holds for the node 32, where all the nodes between 6 and 13 would have been erased.

7.2. DFER, DFERX, GFER, GFERX.

Except for a few details, these four subroutines do the same work. For example, DFERX $[I, J]$ is applied to a structure $J$ that has to be moved to the right under a dominating node $I$. As illustrated below, DFERX $[31, 30]$ moves the structure headed by 30 to the right under the node.
Fig. 6b. Corresponding printed tree

**Figure 6b**

**Pile**

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
LE N V LE N V LE N V LE N V
NO POP ADJ DE NO POP ADJ DE NO POP ADJ DE
HUM ANM IMP PRT HUM ANM IMP PRT HUM ANM IMP PRT
FEM IMP R +CRC FEM IMP R +CRC FEM IMP R +CRC
PLU NMA PRZ A PLU NMA PRZ A PLU NMA PRZ A
PRO PER AN, PRO PER AN, PRO PER An, PRO PER An,
II IN II IN II IN II IN
NUM IPF -DOG NUM IPF -DOG NUM IPF -DOG
ON OST CAU ON OST CAU ON OST CAU ON OST CAU
GEO EXT BUT GEO EXT BUT GEO EXT BUT GEO EXT BUT
FEM NMA MAN FEM NMA MAN FEM NMA MAN FEM NMA MAN
PLU NMA PLU NMA PLU NMA PLU NMA PLU NMA
INF INF INF INF INF
```
Fig. 7a. Sample tree before OTE and NEWR apply.
Fig. 7b. Sample tree where one and newtree have applied
31. (Node 31 was created by rule T2 and DFER was applied on the resulting tree)

**DFER** \([I, J]\) makes node \(J\) the next younger brother of node \(I\)

**DFERX** \([I, J]\) makes node \(J\) the youngest son of node \(I\)

**GER** \([I, J]\) makes node \(J\) the next older brother of node \(I\)

**GERX** \([I, J]\) makes node \(J\) the oldest son in node \(I\)

The general technique for these four subroutines is the following. Before modification, the tree is copied in another area of memory. All the terminal nodes identified with the structure \(J\) take the place of the terminal nodes identified with the structure \(I\). Then, the terminal nodes of \(I\) are requested in the copied structure and parsed with their dominating nonterminal nodes at the right place. **GER** permits the new numbering of the sequence and, if necessary, prunes the tree.

In the example illustrated below (Fig. 9a and 9b), **GER** \([14, 13]\) is applied and node \(CPL\) \((13)\) has been attached to node 16, the father of node 14. If **GERX** \([14, 13]\) had been specified, node \(CPL\) would have been attached directly to node 14, rather than the father of node 14.

7.3. **INTERV**.

This subroutine is used for the permutation of 2 structures. For example, **INTERV** \([I, J]\) where \(I = 24\) and \(J = 28\) gives rise to the structural change illustrated below.

7.4. **INSERT**.

This subroutine is used for the insertion of a new terminal node; for example, **INSERT** \([4, 1HE, 1HT, 1HR, 9]\) introduces node with name **ETR** which becomes a new son of node 9.

7.5. **Other subroutines**.

There is a number of other subroutines concerning conditions specified within a rule, such as the presence or absence of a node or of a feature in the whole structure, the logical restrictions of loose equality or strict equality.
Fig. 8a. Sample tree before GFER applied.
Fig. 8b. Sample tree where GSA has applied...
Fig. 9a. Input structure
Fig. 9b. Output structure after over has applied
Fig. 10b. Output structure after rewrite has applied

OBLING: A TESTER FOR TRANSFORMATIONAL GRAMMARS

ABBRE SYNTAXIQUE 3IT.15 CYCLE = 21

2 - 136 - 2

FIGURE 10b

Fig. 10b: Output structure after rewrite has applied.
Fig. 11a. Insertion of a node (before)
Fig. 11b. Insertion of a node (after)
8. CONCLUSIONS

OBLING is a system which has been implemented in low-level FORTRAN IV for the CDC 6400. It occupies 55,000 60-bit words of memory. It has about 7000 lines of comments and instructions.
REFERENCES

N. CHOMSKY, Aspects of the Theory of Syntax, Cambridge (Mass.), 1965.

A. DUGAS, et al., Description syntaxique élémentaire du français inspiré des théories transformationnelles, Montréal, 1969.

J. FRIEDMAN, A Computer Model of Transformational Grammar, New York, 1971.

D. LIEBERMAN, (ed), Specification and Utilization of a Transformational Grammar, Yorktown Heights (N.Y.), 1966.

D. L. LONDE, W. J. SCHOEHE, tgm: A Transformational Grammar Tester, in Proc. Spring Joint Computer Conference, Part I, 1968, pp. 385-393.

R. PETRICK, A Recognition Procedure for Transformational Grammars, Ph.-D. Dissertation, Cambridge (Mass.), 1965.

J. R. ROSS, A proposed rule of tree-pruning, NSF-17, Computation Laboratory, Harvard University, IV (1966), pp. 1-18.

A. M. ZWICKY, J. FRIEDMAN, B. HALL, D. E. WALKER, The MITRE Syntactic Analysis Procedure for Transformational Grammars, in Proc. Fall Joint Computer Conference, Vol. 27, Pt. 1, 1965, pp. 317-326.
