II - DISCUSSION ABOUT SOFTWARE TOOLS FOR M.T. AND NATURAL LANGUAGE PROCESSING.

Bernard Vauquois

Groupe d'Etudes pour la Traduction Automatique
Universite Grenoble
France

<1>

Content

1 - THE PERIOD OF CONTEXT-FREE PARSERS AND SIMILAR ALGORITHMS

In a second generation system, the process of translation is realized by 3 sequential periods : analysis, transfer, generation. Also, the linguistic data are separated from the programmed algorithms. According to this scheme, the linguistic data appear as dictionaries and grammars which are written by using an adequate formalism provided by the software tools. It has been seen that, usually , each period was divided into 2 steps :

Analysis : { morphological analysis / intermediate level analysis

Transfer : { lexical transfer / structural transfer

Generation : { syntactic generation / morphological generation

In the early experiments, and sometimes now, some steps are avoided. That was frequently the case for morphological analysis of English because the different forms coming from the same lexical unit are a few. The price which has to be paid for eliminating a morphological analysis grammar, is a larger size of the dictionary and the time spent for the coding of 2 or 3 times more items than necessary.

The syntactic generation is also avoided when the intermediate structure matches the level of surface syntax ; in such a case, the structural transfer furnishes directly the desired result of a syntactic generation. Rightly or wrongly, the consequence in the 60's has been a concentration of efforts and cleverness to find out efficient algorithms for "parsing". In many cases, the parsing was bounded at the level of a syntactic analysis.

In most of the laboratories these parsers were based on a Context-Free model and for 10 years or more top-down vs bottom-up techniques were competing for better algorithms.

計算言語学 18－2
(1979. 7. 11)

For some other laboratories, a dependency model was preferred to the C-F model, and for the Philadelphia School it was a string adjunct grammar. All these techniques aimed equivalent levels of interpretation , even if their authors were convinced of doing different linguistic approaches.

In every case the input string of words of each sentence was replaced by a labelled tree whose interpretation was a surface syntactic structure.

In every case, also, the algorithm came from automata theory where the main goal is the detection of well-formed sentences. In other words, the parsers are "acceptor automata" which have to answer by yes or no about any submitted string, candidate for a sentence. The tree structure which is deduced in case of a positive answer and which is the most interesting result, is, in fact, a by-product.

It is obviously true for a C-F model since many different C-F grammars can define the same language ; consequently, the same sentence will be assigned different trees according to the selected grammar.

For the other cases, the resulting structure depends also on the choice which has been defined to write the dependency rules or the adjunct strings.
Unfortunately, in all cases, this choice is not guided by the desired result but more or less imposed by the algorithm ; indeed, for a predictive analyzer the C-F rules must be written in Greibach's standard form ; on the contrary a Cocke's algorithm imposes a Chomsky's standard form grammar. Earley's algorithm offers more flexible possibilities.

2 - REMARKS ABOUT PARSERS

First of all, the existence of parsers does not solve all what is needed as software tools for a translation process. Even, if dictionary look-up algorithms are included in the whole system, the structural transfer cannot be handled. Furthermore, if the intermediate structure is expected at a deeper level of interpretation some other tool is necessary to transform the results of the surface analysis. Also, it may be requested that some deep analysis could be obtained without parsing through the surface structure ; indeed, sequential levels of interpretation have the great disadvantage to carry on a lot of ambiguities which cannot be solved at the earlier stages.

Second , we have seen that the obtained structure was a by-product which is not always compatible with a consistent linguistic interpretation. For instance, as soon as 1965, Susumu Kuno at Harvard University needed to append some complement to each rule of the grammar which was written for the predictive analyzer, in order to build step by step an other tree structure besides the tree coming from the derivation.

Third, such grammars are defined by the total set of rules. If the grammar is not restricted to the definition of a small sub-language and is expected to cover large quantities of various texts in a given natural language, then the number of rules becomes extremely important. Under such conditions, to debug a grammar containing hundreds of rules is an impossible task. Every time a rule is modified in order to accept new sentences, this modification may prevent to recognize former sentences which were accepted.

A long experiment, we have done in 1966-1970 on analysis of Russian stopped after 4 years of improvement ; it was impossible to analyze more than 70 % to 75 % of the sentences by such a grammar. For practical reasons (and that is not incompatible with theoretical methodology) it is highly desirable to work with modular grammars.

Fourth, in case of failure for a given sentence, an empty result is obtained for that sentence. It is extremely unpleasant to block the translation process for such sentences because the formal grammar is not totaly adequate or the sentence itself is not perfectly well-formed. For practical reasons, too, a translation should be obtained even for badly written input texts.

Fifth, such grammars can be considered as static sets of rules. The grammar, by itself, has no effect on the way how to proceed with the rules. Consequently, the algorithm of the parser is a combinatorial one, in order to detect all possible structures. The only possible restriction against the combinatorial explosion comes from the algorithm itself.

the linguist cannot indicate through the grammar rules how to speed up the process. Another consequence of the combinatorial effect is the creation of parasite ambiguities.

3 - THE NEW SOFTWARE TOOLS

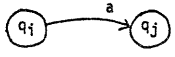
At the early 70's all the preceeding remarks were vaguely perceived, but certainly not in clear way enough to make an opportunity of solving all the negative aspects at the same time. Nevertheless, MIND-system (1970) by M. Kay and R. Kaplan at Rand, Augmented Transition Networks (1970) by W. Woods at B.B.N. , Q-systems (1971) by A. Colmerauer at Montreal, ATEF (1972) and CETA (1974) by J. Chauché at Grenoble, PLATON (1974) by M. Nagao and J. Tsujii at Kyoto, REZO (1975) by G. Stewart at Montreal, TRANSF (1976) by P. Guillaume et M. Quézel-Ambrunaz, ROBRA (1978) by C. Boitet, and SYGMOR (1978) by D. Jaeger at Grenoble, all these software tools show how dense has been this activity during this decade.

≪3≫

3.1. A survey of different options

The ATN presented by W. Woods is strongly oriented by the tradition followed at Harvard University for many years. The system goal is the production of a labelled tree (or several trees, in case of ambiguity) to be associated to any input sentence. That can be considered as the summit of parsers. The main idea, very fruitful, in ATN is the use of the finite state graph for the representation of a process which recognizes languages more complex than finite state languages.

In a finite state language representation by means of a finite graph, the elementary scheme is the following :



where the nodes q_i and q_j represent the states and the label "a" on the arc means the reading (in recognition) or the writing (in generation) of the terminal symbol "a".

In ATN the label borne by the arc is more generally an "action". This action may be the reading of a terminal symbol as a particular case ; It may be also the transfer to another initial state (whose name is the label of the arc) with the preservation of state q_j in a push-down store. Considering that any transition rule, from one state to another, has a choice among different actions, may test for conditions and build a tree structure with any number of registers, the ATN is a tool which meets some of the expectations mentionned in the preceeding remarks about parsers.
Indeed, the desired tree structure is a definite goal and does not appear any longer as a by-product of a derivational grammar ;
Also, the scattering of a large grammar in a state-graph network, where sub-networks are easily referenced is a way for modularity requirement.
Nevertheless, the ATN is a string to tree transducer, having necessarily a left to right scan, inspired by the predictive analysis. It is also an acceptor automaton, as far as the requirement for a final state is the only criterion to get a result. As the input data structure is not homogeneous with the output data structure, it cannot be used for a tree to tree transduction or a string of trees to a string of trees transduction . Finally , as ATNs have the power of Turing machines, the halting problem is not decidable.
MIND system and Q-System share many features. They are both transducers for strings of trees to strings of trees; every string of occurrences is considered as a string of degenerate trees. The advantage of such systems is that only one tool is able to support any step of the translation process. Also, both systems have a pattern matching procedure incorporated in their processors. Q-System is certainly easier to learn than MIND-System ; but in counterpart, the user may control the flow of process in a more efficient way with a MIND-system than with a Q-System which is more combinatorial. We can see, however, the first attempts of introducing heuristics within a non-deterministic transducer.
As the input and output data structures are homogeneous, the modularity of large grammars may be obtained by a sequence of smaller grammars. Both systems have the power of Turing machines with the same consequence for the halting problem.

PLATON is based on ATN-model and has various additional capabilities such as pattern-matching, flexible backtracking, and so on. A specific feature of PLATON is the ability to link the internal flow of the algorithm with arbitrary LISP functions (defined by the user) which behave as procedures

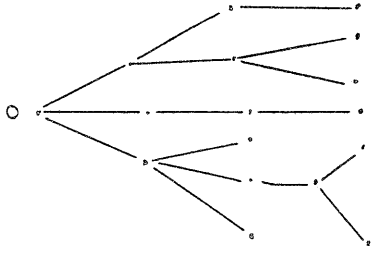
on the selected substructures. It seems that PLATON was designed for analysis purposes ; but, in order to avoid the left to right scan of a string, it has been paid attention to the simultaneous occurrences of lists (strings) and trees. The consequence is the handling of a data structure which is equivalent to MIND and Q-Systems. The difference lies in the fact that for Q-Systems strings and trees are considered both ordered, whereas in PLATON lists are ordered and trees are not. REZO is an improvement of ATN which has been realized for the TAUM translation project. It remains, nevertheless, a string to tree transducer and cannot be used for other steps besides the intermediate level analysis.

The different components which have been realized at Grenoble are devoted to different steps of the translation process.
Indeed, a single tool, like Q-System for example, is an attractive feature ; for linguists who have only a very weak experience in computer science, it is a tool easy to learn, with which all grammars and dictionaries may be written for any step of the translation process.
Nevertheless, such a tool was estimated unadequate to process large scale models. Also, only one tool for doing everything means that this tool is able to perform the most simple tasks as well as the most sophisticated ones. Consequently, the tool needs to have the maximum of power and it may be unefficient for the performance of the most elementary steps (dictionary look-up and morphological analysis or generation). It was the reason which has been evoked to build a morphological analyzer (non deterministic) on a finite state model (ATEF) ; a tree to tree transducer (first CETA, then ROBRA) for intermediate level analysis, structural transfer, syntactic generation, with a sub-recursive power in order to be sure of the decidability of the halting problem TRANSF for bi-lingual dictionaries processing ; finally a morphological generator (deterministic) on a finite state model (SYGMOR).
In fact, ATEF and SYGMOR should be gathered together in a single component with a mode parameter (deterministic or not).
Also, it is expected to find a single external language for all components in order to gain the advantages of a unique tool ; the different components will be still remaining in specific families of algorithms in order to preserve the maximum possible efficiency.

3.2. Heuristics in language processing

In first generation M.T. systems, there is no separation between linguistic data and algorithms. The detection of linguistic phenomena (lexical ambiguities, functional ambiguities, relation between words, etc...) and their processing are directly programmed by means of a programming language often a general purpose programming language (assembler code, FORTRAN, COBOL, . This way of proceeding is one of the reasons which bound strongly the improvement in first generation systems. With the direct programming of M.T. system, the programmer-linguist has at his disposal the total flexibility for managing as he wants. But very soon, the programs become inextricable.
Just the opposite of that method is the generation of Context-Free parsers and similar routines.
The grammar is a set of rules written according to same agreed formalism.
The dynamic process is a general routine which uses such a grammar as a parameter. The linguist cannot control (or sometimes just a little) the combinatorial effect of the general algorithm.

With the new wave of tools, it seems that some attempts have been developped progressively in order to introduce more and more facilities for the linguist to control the flow of a predetermined algorithm. This kind of approach has to be careful if a turn back to the first generation (at a higher level of programming) needs to be avoided. That was sensitively the case of such a possible way with the PROGRAMMAR language designed and used by T. Winograd as an extension of LISP.
So, the problem is how to complete linguistic data (dictionaries and grammars) by adding a "strategy". Given a combinatorial procedure, a first idea consists in interrupting the search for a solution as soon as it has been discovered it was a wrong way and in backtracking immediately. Any non-deterministic procedure follows a decision tree where the leaves match the end of a path to accept or to reject an expected solution. In many cases, however, a failure



may be detected before the end of a path. Also, an efficient heuristics tries to give precedence to the paths which have the highest like lihood to lead to successful trials. this facility should be connected with the possibiiity of erasing some waiting branches of the decision tree as soon as the corresponding information is found in some node.

This kind of strategy is offered in ATEF, for morphological analysis, to the linguist :

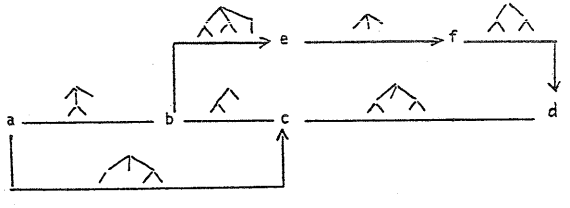
- organize the dictionary entries in such a way that the most frequent segments are processed first
- insert some special functions in the grammar rules to erase partially or completely the remaining branches of the decision tree.

Many other aspects of heuristics appear in a more complex way through tree to tree transducers.

3.2. Tree to tree transducers

Under this title are considered also string of trees to string of trees transducers ; therefore, the main systems now in use are MIND (not quite sure), Q-systems, PLATON and ROBRA.

A string of trees can be considered as a single tree with an additional node which is the new root of this single tree. The matter is not so simple when the transducer is dealing with a set of strings of trees. That is the case of a Q-graph.



where the different strings of trees follow the paths :

{ a b c d
{ a c d
{ a b e f d

Such a Q-graph is an alternative to represent the ambiguities which appear during the analysis.

A - Pattern matching ; process

In all cases, the application of a transformation rule comes from a pattern matching. Two kinds of approaches for processing may be considered

a) the process is sequential ; in such a case, the conditions given by the left part of the rule are the only restrictions to be satisfied for the application of the rule. In other word, any candidate rule is applied as soon as the pattern matching conditions of its left part is fulfilled.

b) the process is parallel : in such a case, the preceding requirements are still necessary conditions but not sufficient any longer. It is possible, indeed, that several candidate rules, which are agreed according to their own conditions, are competing to be applied on overlapping patterns in the object structure. Certainly, it could be anticipated a non-deterministic procedure by means of which all the different incompatible applications would open new branches of the decision tree. It has been preferred, in CETA and then in ROBRA, to avoid such a combinatorial explosion and supply the computational linguist with decision rules in order to solve the alleged conflict between transformation rules. For example, decision rules of that sort may be ordered choices between :

- innermost versus outermost pattern
- leftmost versus rightmost pattern
- the highest precedence between rules, and so on.

That is one of the many aspects of introducing heuristics (or strategy) into tree to tree transducers.

B - Pattern matching ; description

This aspect concerning how to write the conditions for a pattern matching may sound somewhat narrow-minded, as long as the same purpose is purchased by any means. However, as different options have been selected

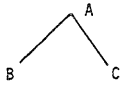
and as different computational linguists had to use either one or the other way of writing for the description of many rules, this becomes worthwile to compare the advantages and shortcomings of these options.
It is sufficient to consider the description of a pattern matching in a tree structure ; indeed, the situation for a string of trees is easily extrapolated.
The requested pattern is a tree structure of nodes which appears as a subtree of the object-tree. Nodes are decorated by labels. Two questions arise right now :

- How many labels are allowed for the content of one node ?
  The answer may be either "only one" or "as many as you wish"
- Is the subtree pattern a total or a partial subtree of the object tree ?
  The answer is "total" in Q-system, "partial" in ROBRA.

The effects of these choices are extremely relevant.
About the number of labels assigned to each node, the influence of the formalized linguistics for 20 years has been very important. That is probably the reason why so many software tools for natural language processing allow only one single label per node. It seems to be a disadvantage because the computational linguist must organize by himself a lot of informations concerning a single linguistic unit, by means of a tree structure. Thesetree structures which have no linguistic meaning are incorporated in the tree structure which is relevant for relationships between words ; one can find there, the source of confusion.

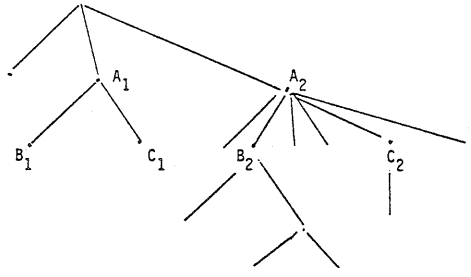
If it is the other alternative which is preferred, then it becomes unpractical to call a node by its label ! Only formal names (node parameters) can be used to describe the tree pattern ; the conditions about labels must be written separately. That is true ; however, parameters are needed in both cases.

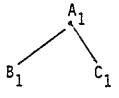
About the choice between partial or total subtree it is a matter of convention.
Assuming that the pattern indicated in the left part of a rule is written A(B,C) given the subtree :

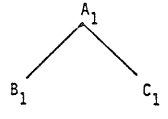
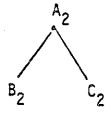


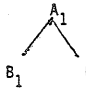
What is the convention for a match in the object tree ?

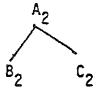
object tree :



Is only the total subtree  is accepted

or both  and the partial subtree 

In the first case the detection of  is very shortly written

but the detection of  has to be written in a more complicated

way, using a lot of list parameters in Q-system :

A(U*,B(V*),W*,C(X*),Y*)

On the contrary, if the second option is choosen the writing of A(B,C) is suitable for both matchings.

If,only  is relevant, it is needed to have a special symbol for the lack

of node. In ROBRA, this symbol is * . The preceding example is then written

A(*,B(*),*,C(*),*)

Depending on the option, attention has to be paid for the transfer function of the nodes.

C - Organization of rules - Halting

Different ways may be proposed to include the transformation rules in a dynamic process.
The most simple, of course, is the total combinatorial way, as it was the case for C-F parsers. That is the option of Q-systems. Indeed, a Q-grammar is processed on an input Q-graph and every rule matching the conditions is applied on the Q-graph which is increasing every time a rule is processed.
The right part of such a rule is also a string of trees which is added on the object Q-graph. The Q-system halts when no more rules can be applied ; in some cases the system is endless.
The convention which holds to clean up a Q-graph leads to keep on the graph only those arcs which have never been used for the application of any rule when the system came to a stop.
As such an output Q-graph is usable as input for another Q-system, the combinatorial effect can be decreased by a sequence of Q-systems with few rules instead of a single one with many rules. However the linguistic constraints do not allow to reduce the size of each grammar as much as it could be desirable.

The MIND system organizes a control of the flow through the rules themselves by a lot of functions. It is no more the best solution because the same rule could have different orientations of strategy at different steps of the process.

PLATON seems to have followed the discrimination between the set of grammar rules and the control mechanism inspired by ATNs ; some improvements have been added to avoid the lack of flexibility in strategic decisions, which characterizes the ATNs.

In ROBRA, as it has been shown in the previous paper, the different notions of "transformation rule" (which may be context-sensitive) "transformation grammar" to build a dynamic module, and "transformation system" to define strategies among the modules, are the basis for discriminating the grammar rules and the control mechanism. Also, the different modes associated to each grammar increases the flexibility.

In conclusion, many common ideas have been developped on this topics. The stress has been put in different parts according to the feeling of involved searchers. The same goal is shared by everyone. Much cooperative work can be done together.

APPENDIX.

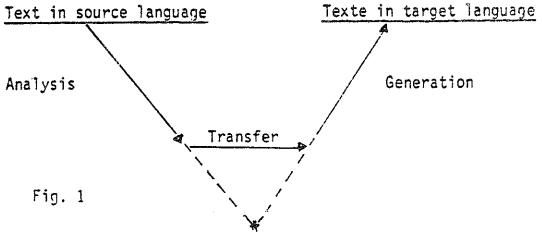


Fig. 1

A translation system is a sequence of six models :

(1) morphological analysis, (2) multilevel analysis for the "analysis phase" (monolingual), (3) lexical transfer, (4) structural transfer for the "transfer phase" (bi-lingual), (5) syntactic generation, (6) morphological generation for the "generation phase" (monolingual).

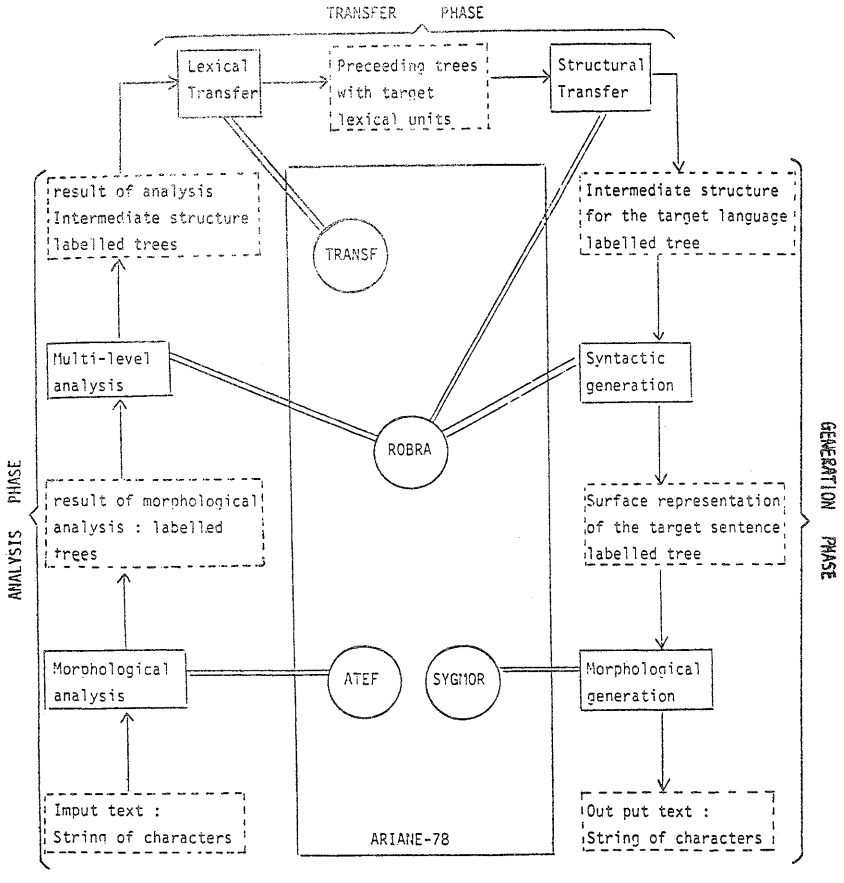


Figure 2

actual Grenoble system.