Computers and the Humanities 20 (1986)
©Paradigm Press, Inc.

Machine Translation and the SYGMART System

P.C. Rolf and J. Chauché

1. Differences between Formal Languages and Natural Languages

Much of the work in the field of machine translation and, more generally, of computational linguistics, is based on work in modern linguistics (Chomsky, Montague, etc.) and in computer science, while differences between natural language, the linguist's area, and formal (programming) languages, the field of computer science, are being disregarded.

Usually this difference is defined simply as the problem of ambiguities: natural languages are ambiguous, formal languages (at least programming languages) are not. We are convinced that the essential differences between the two types of languages are different and that ambiguities are one of the manifestations of these differences and not the differences themselves.

1.1 *Natural Languages Are Not Precise*

Every linguist knows from experience that, when he has constructed a grammar for a subset of a natural language, there will always be cases which are acceptable, but which are not described by his grammar. Computational linguists, who are usually less interested in generating than in analyzing, generally design grammars which from a generative point of view do both too little (not all cases are generated) and too much (a number of cases are generated which are not looked upon as "grammatical"). The latter is not a problem: they only offer correct sentences for analysis.

Jacques Chauché is a professor in computer sciences and collaborates with Professor Bourquin, Celta, Nancy. P.C. Rolf teaches linguistics at the Catholic University of Nijmegen (Holland).

Furthermore, in natural language one often comes across constructions (for instance adjuncts) which can occur in various positions in sentences (e.g., "he went home yesterday" vs. "yesterday he went home"). So the order of sentence constituents cannot in all cases be defined precisely.

1.2 *Natural Languages Are Not Defined*

In contrast to programming languages, natural languages are not defined by formal grammars. This makes the linguist's job completely different from that of a computer scientist. Linguists design models for existing languages which are not defined and are not even known in a less formal sense: linguists are always debating the grammaticality (acceptability) of sentences. This problem does not exist among computer scientists: a grammar is designed and this grammar determines whether or not a "sentence" is grammatical. Computer scientists may have problems with the defined language in the sense that this language does not express adequately what they want to define or the language they have in mind is not defined at all. The grammar defines precisely the set of correct sentences.

Perhaps linguists do not know their language; on the other hand they know quite precisely large series of local phenomena.

1.3 *The Notion "Sentence"*

For linguists the notion "sentence" is linked to a number of demands that are put with respect to well-formedness. This provides the possibility of disregarding a multitude of phenomena, such as elliptical sentences (e.g., "Yes, it is." or "And for me one of those delicious red ones that you told me about yesterday"), while on the other hand it gives rise to many problems, as in the case of coordination and concatenation of sentences.

Whoever occupies himself with machine translation, and wants to be able, eventually, to translate normal, running text, will often have to look beyond sentence boundaries.

Writing grammars for texts seems an impossibly difficult task. But here, too, the following applies: a number of relations are known, and it should be possible to describe these relations between parts of different sentences in a formal way. For a start, just for the correct translation of pronouns with antecedents in previous sentences, it is necessary to be able to go beyond sentence boundaries.

2. Machine Translation: A Different Occupation for Linguists

Machine translation, as has been mentioned, is a different occupation for the linguist than the development of generative models of any natural language. The linguist has to define the transition from one (natural) language into the other. This is not the same as building an analyzing grammar and, based on the results of this grammar, a generative one.

In these activities he must have the possibility, first, to be not precise but vague, wherever this should be necessary; second, to be able to restrict himself to local structures (e.g., NPs) and not always have to consider complete sentences, and thirdly, wherever necessary to consider global structures (e.g., pronoun reference) and not to have to restrict himself to (complete) sentences.

Indeed, these demands are not so strange for structuralists: it is perhaps peculiar that modern linguists are not aware of this. In the brief space of this section we discuss three items which are relevant in this respect.

2.1 *Chomsky grammars*

All the grammars in the Chomsky hierarchy of formal grammars (Chomsky, 1959) define a set of strings. It has already been said that for a natural language this may be too precise a means, since the set is not exactly known. In the case of machine translation it is evident that word order changes, deletions and insertions will have to take place, in short, that transformations are necessary. But the transformational grammar in the Chomsky hierarchy (Type 0) cannot be used as an analyzing grammar (see 2.3).

There is yet another objection against Chomsky grammars, viz., the notion label. Concord of gender, for example, and number and case between subject and predicate are extremely difficult to indicate by means of these labels as they are found in many languages. To express these matters, one needs a great extension of the number of labels and, coupled to this, an extension of the number of rules. The resulting lowered degree of abstraction and grammars is difficult to grasp.

Another problem of Chomsky grammars is the expression of endless nesting and endless coordination. Both can be expressed only by recursion, so one has no means of discriminating between the two. A further problem is that when applied as analyzing grammars they can either accept or reject sentences. When sentences are accepted, then as a rule the derivation(s) (treestructure(s)) is (are) produced by the parser. For a translation these trees could be a starting point (Van Bakel, 1984 is a good example), but the other objections remain.

In machine translation we do not wish to be able to reject anything at all: the input presented is regarded as belonging to the language and has to be translated. In computational linguistics this is normal practice, but one does not take advantage of this principle.

In computer science some types of grammars have been defined with respect to a number of the problems mentioned, such as two-level Van Wijngaarden grammars, Affixgrammars, Attribute grammars, ATNs, etc. (for some of these and others see Winograd, 1983). As far as linguistic activities are occupied with precise subsets of natural language, they are certainly worth studying, but they are not the most suitable for machine translation. They all remain too precise.

2.2 *Semantics and the problem of ambiguities*

The greatest problem in machine translation, from the side of Artificial Intelligence often said to be the only problem, is usually said to be the "knowledge of the world." Scripts and/or concepts are proposed as a solution to this problem. Without denying that on this level, too, there are a great many problems, we are strongly convinced that the precise extent and nature of this problem cannot be known until series of linguistic and algorithmic problems have been studied in an adequate way. The "knowledge of the world" problem is strongly

The "knowledge of the world" problem is strongly connected with the problem of ambiguities. It is remarkable that ambiguous sentences are far less frequent than is sometimes suggested: almost every one always comes up with the same examples and in a wider context these examples are hardly ever ambiguous. In other words, if one wants to translate a text, one is confronted with local, or rather, in terms of the process of machine translation, "temporary" ambiguities. (Whenever sentences are meant to be ambiguous, one normally cannot translate them at all. So, not everything can be translated.)

If in the course of the translation process possibly correct sentences are constructed on the basis of all those "temporary" ambiguities, this leads to the combinatorial explosion (Barr, 1981) that has thwarted many a research project. For the linguist, moreover, this way of working implies that he must write grammars that are precise enough to reject ungrammatical (unacceptable in a more intuitive meaning) sentences as well: he must define precisely all the sentences of the language and none other than these (our greatest objection with respect to natural language processing).

Our conclusion is that as far as the analysis of the source language is concerned, a method based on possible sentences is not the right one, but a method based on sub-analyses that are really certain must be applied.

2.3 *Algorithmic notions are linguistic notions too*

In designing grammars of natural languages most linguists have hardly been concerned with algorithmic matters with respect to the application of these grammars. In fact, they define their language statically, and, as has been said earlier, precisely. The assumption that a parser generator for their grammars is sufficient to yield results analytically seems theoretically correct, and turns out to be incorrect only from an algorithmic viewpoint. Type 0 grammars do not terminate at incorrect input or in search of alternatives.

Computational linguists have for some time been confronted with algorithmic aspects of language definitions: even in writing context-free grammars they often have to reckon with matters like left- or right-recursion, because of the application of these grammars. In more powerful grammars, certainly in type O, algorithmic aspects have to be reckoned with to a far greater extent: one has to write for the same set of phrases quite different grammars depending on the use of the formalism, e.g., Two level Van Wijngaarden Grammar or EAG or Type 0 (Chomsky).

When grammars are used to construct sub-analyses, without the entire input being taken into consideration, the linguist will have to be able to express algorithmic notions with respect to where in the input vagueness must be exercised and where precision, which (parts of) grammars and rules must iterate or not, and where recursion must be applied.

3. SYGMART: a translator generator

The SYGMART system presents the linguist with many possibilities to define translation systems (Chauché et al., 1982). These translation systems operate on the basis of tree transformational grammars linked to one another in a conditional network. In this section some essential features relevant with respect to the foregoing, will be discussed.

The transformation of one tree structure by means of a succession of transformations of subtrees is central in SYGMART. This is accomplished in the subsystem TELESI. Since SYGMART is a translator generator for natural language (strings), both a morphologically analyzing component (called OPALE) and a morphologically generating component (called AGATE) have been provided. The former turns in input text (string), into a tree, and the latter a tree into an output text (string), using grammars and lexicons. These two subsystems are left outside the scope of this paper, as are scores of other facilities in TELESI. We shall restrict ourselves to the most relevant features of the tree transformational component TELESI.

3.1 *The notions tree and label*

At the user's choice, TELESI uses one to maximally sixteen trees (Chauché, 1984), in which initially every tree contains the input at the leaves, according to the conventions of OPALE. Since all the trees entirely or partly contain the same nodes, it is possible to define far more complex graphs on the input than a simple tree would permit. So in fact there are two sets: a set of points and a set of (maximally sixteen) trees of those points, and not all the points need occur in all the trees.

Furthermore, there is a set of complex data (instead of the traditional label) and a set of functions from the points to the complex data: for every point there exists a function to an element from that set of complex data. It is not necessary for them to be a unique element from that set for every point. In a simple way several points can be made to have the same complex data as their values. This is essential where co-references in natural language are concerned and matters like Trace and Pro in Chomskyan theory (Chomsky, 1981).

So abstractions have been made from the notion label as it is used by Chomsky. In the output of SYGMART the trees are written in a bracket structure and the points are numbered (these numbers are written down as labels, as it were); the numbers refer to complex data, also numbered. Nodes with identical numbers have the same complex data as their values.

3.2 *Subtrees and tree pattern matching*

Each one of the maximally sixteen trees of TELESI is transformed by successive transformations of subtrees. A subtree consists of an arbitrary point with all its descendants.

The SYGMART user defines transformations by means of matching schemes (optionally followed by conditions with respect to the node values), followed by an arrow (=)), followed by a transformation scheme. In this the rule holds that identical names in the structural description of the matching scheme (to the left of the arrow) and the transformation scheme (to the right) represent the same nodes, in which nodes that are mentioned on the left, but not on the right, are deleted, while nodes that are mentioned on the right but not on the left are new nodes.

In this framework the mechanism to write down structural descriptions is essential. Traditionally we have only brackets ('(' and ')') to indicate dependency. In SYGMART the question mark ('?') has been added, which, in front of an opening bracket indicates a possibly indirect dependency. Sisters are separated from each other by a comma (','). To indicate horizontal continuity (vertical continuity is the traditional bracket) the asterisk ('*') is used. In fact, the asterisk stands for nothing, neither horizontally nor vertically. In this it is implied that *(0) indicates a top, and 0(*) a leaf, while 0(A,*,B) indicates that A and B are direct sisters.

Furthermore there are the operator [illegible], which indicates an optional point, the operator [illegible], which indicates a possible inverted order on the same level, and the operator '@', which defines a structural context.

The definition of an elementary transformation, according to the principles described above, is used to construct an elementary iterative or recursive grammar with transformational power. A recursive network constructed from a set of such elementary grammars defines a TELESI grammar.

It is obvious that the linguist has all the possibilities to remain vague or to be precise, wherever this is necessary in the structural description of his transformations.

It has already been said that the notion label has been defined in a more abstract sense in SYGMART than in Chomsky grammars, viz., as a function from a point to an element of a set of complex data. In the transformation rules there are ample possibilities for positive or negative tests on subsets of values of these complex data, so that here, too, there are a great many possibilities for precision and vagueness. With regard to words belonging to various categories, for instance, it is not necessary to construct different sentences on the basis of this fact. So one has the means to avoid combinatorial explosion.

Precisely because there are possibilities for vagueness, the linguist will have to know how the subtrees are searched for and transformed in TELESI. One has to know for instance that TELESI is working in a parallel way on two levels. Thus he will have to reckon with two types of parallelism: the TELESI trees are transformed in a parallel manner and for each tree the rules are one by one applied in a parallel way on all disjunct subtrees. In this the linguist must be familiar with the workings of tree pattern matching: a matching scheme is searched for from right to left and within a subtree found, the one occurring on the highest level.

As a consequence, structures called recursive in linguistics have to be written down in an iterative way, while matters such as coordination and concatenation can be written down in an easy way, applying some rules, usually three, one after the other.

The knowledge base which is associated with the TELESI grammar defines a last characteristic of the system. Consequently a TELESI grammar is dependent in its internal structure on the input presented. Elementary grammars change their behavior in function of the input and its consequences. This approach is not static but dynamic because the input of the knowledge base is built in dynamically, based on the input of the SYGMART system and its transformations.

The SYGMART system has three basic aspects. First, the transformation of an abstract unlabeled structure, second, the evaluation of the multiple labels and third, the consultation of the possibly large knowledge base. These three aspects are in fact mixed and influence each other.

3.3 *An example*

The simple example presented here uses only one of the sixteen possible trees. The TELESI grammar (all the rest is left out) defines the transformation of Dutch number names up to one million (the grammar is designed purely for demonstration purposes, so the extent of the lexicon is arbitrary) into the corresponding figures and is derived from Rolf, 1983 and Rolf, 1985.

```
  &GRAMMAIRE.
&ENTREE: TELWRD(I).
    DELTIGEN:   O(1(*),*,2(*))     /
             2: ((VORM=´tig´)|(VORM=´en´))
    =>  O(1)     /
             1: 1(<VORM(2)=´tig´:SOORT=SOORT(2)>).

    KEEROM:  O(1(*),*,2(*))  /
             1: SOORT=EENHEID ;
             2: SOORT=TIENTAL
    =>  O(2,1).

    DEL_ENHO: O(1(*),*,2(*),*,3(*)) /
             1: (SOORT=DUIZTAL);
             2: (SOORT=EENHEID);
             3: (SOORT=HONDTAL)
    =>  O(1,3) /
             3: 3(REPR=REPR(2)).

    VULOP: O(1(*),*,2(*))  /
             O: (WRDSOORT=TELW)    /
                 AANW(1,2)
    =>  O(1,3,2)  /
             3: (KENTEL(1)).

    VULAAN: O(1(*),*)  /
             1: (SOORT=DUIZTAL)|(SOORT=HONDTAL)|(SOORT=TIENTAL)
    =>  O(1,2)  /
             2: (KENTEL(1)).

    DEL_DU(*(*);O(2)): O(1(*),*,2(*)) /
             1: (SOORT=EENHEID)|(SOORT=COMBI);
             2: (SOORT=DUIZTAL)|(SOORT=HONDTAL)
    =>  O(2) /
             2: 2(REPR=REPR(1)).
-->  STOP.
  &FIN.
```

The input tree for this grammar consists of one or more number names, all of them in disjunct subtrees. The leaves of these subtrees contain the lexical parts of the number names.

The grammar TELWRD is an iterative one (indicated by I), which means that the rules are applied iteratively one by one in the order presented here, until no rule can be applied anymore. The first rule, DELTIGEN, deletes all occurrences of 'tig' and 'en' (English 'ty' and 'and' in number names like 'a hundred *and* forty') and next KEEROM changes the order of tens and units, which are normally inverted in Dutch. The other four rules (DEL ENHO, VULOP, VULAAN and DEL DU) see to the insertion and addition of zeros (e.g., in number names like 'a hundred' and 'one thousand and two') and the contraction of units and thousands or hundreds ('duizend' ['a thousand'] is 1000 and 'drieduizend' ['three thousand'] is 3000 and not 31000). Note that the grammar does not define the language of Dutch number names, but performs the correct translation of all number names into figures.

Vagueness is exercised here so that even number names that are incorrect (not well-formed), but are normally well understood, are translated correctly. For example 'vierentwintighonderd-duizend' (English: 'twenty-four hundred thousand') is not correct, but is translated by this grammar into 2400000.

4. Conclusion

Machine translation and, more generally, natural language applications in computers demand much more complex analyses, both from a linguistic point of view and from the point of view of computer science than those which are normally used in the field of artificial intelligence.

The local approach of the analysis concerning the topology of the terms and not restricting the domain of the application, which is opposed to the majority of today's applications, has the following advantages:

1) TELESI grammars are defined for all input cases and thus always produce output.
2) They do not presuppose to know everything about the languages concerned. Even a TELESI grammar does not obligatorially define the recognition of a well-defined subset, but it defines an application of the set of all phrases.
3) Obviously it is always possible to define a supplementary constraint in addition and so to construct all different classic applications (e.g.: the definition of a context-free grammar is always possible in a TELESI grammar, as is Towers of Hanoi).

The technical implications of the SYGMART system are pointed out in Chauché, 1974 and, somewhat differently, in Boitet, 1976. An application with respect to the analysis of French, implemented in a predecessor of SYGMART, is published in Weissenborn, 1978.

REFERENCES

{Bakel, 1984] Bakel, Jan van, *Automatic Semantic Interpretation*, Dordrecht, 1984.

[Barr, 1981] Barr, Avron and Edward a. Feigenbaum, *The Handbook of Artificial Intelligence*, Stanford, 1981, Vol. I.

{Boitet, 1976] Boitet, Christian, *Un essai de réponse à quelques questions théoriques et pratiques liées à la traduction automatique, définitions d'un système prototype*, Thèse d'Etat, Grenoble, 1976.

{Chauché, 1974] Chauché, Jacques, *Transducteurs & Arborescences, Etudes et réalisations de systèmes appliquées aux grammaires transformationnelles*, Thèse d'Etat, Grenoble, 1974.

{Chauché] Chauché, J., *Système de manipulations algorithmiques et récursives de texte. SYGMART*, Laboratoire de traitement de l'information, Le Havre, France.

[Chauché, 1982] Chauché, J., V. Cheboldaeff, R. Lescoeur, M. Jatteau, *Rapport de présentation du project relatif à l'écriture des spécifications d'un système de traduction assistée par ordinateur*, SLIGOS, Conseil, assistance et réalisations, produits informatiques, services de gestion, Avril 1982.

[Chauché, 1984] Chauché, J., *Un outil multidimensionnel de l'analyse du discours*, in: Proceedings of Coling84, Stanford University, California, pp. 11-15, 1984.

]Chomsky, 1959] Chomsky, Noam, *On Certain Formal Properties of Grammars*, In: Information & Control, 1959, 1, 91-112.

[Chomsky, 1965] Chomsky, Noam, *Aspects of the Theory of Syntax*, Cambridge Mass, M.I.T. Press, 1965.

[Chomsky, 1981] Chomsky, Noam, *Lectures on Government and Binding*, Foris Publications, Dordrecht Holland, 1981.

[Rolf, 1983] Rolf, P.C., *Vertalen van getalsnamen*, Verslagen Computerlinguïstiek No. 3—1983, KU Nijmegen.

[Rolf, 1985] Rolf, P.C., *Beschrijving van het vertaalsysteem SYGMART, versie 4*, intern verslag KU Nijmegen, 1985.

[Weissenborn, 1978] Weissenborn, Jürgen, *Eine Basisgrammatik für die automatische Analyse des Französischen im Hinblick auf die maschinelle Uebersetzung*, Dissertation, Saarbrücken, 1978.

[Winograd, 1983] Winograd, Terry, *Language as a Cognitive Process*, Volume I: Syntax, Addison-Wesley Publishing Company, Massachusetts, 1983.

For five years both authors have been working together in the field of machine translation, the former as a linguist at Nijmegen University in the Netherlands and the latter as a professor of computer science and a member of the CELTA, Centre d'Etudes Linguistiques pour la Traduction Automatique, Nancy, France. In this article they point out some basic ideas that underlie the translator generator SYGMART. In view of the state of affairs in linguistics this article was not (yet) translated by SYGMART, but by Pieter de Haan, to whom the authors are very grateful.