COLING 82, J. Horecký (ed.)
North-Holland Publishing Company

MACHINE TRANSLATION BASED ON LOGICALLY ISOMORPHIC MONTAGUE GRAMMARS

Jan Landsbergen

Philips Research Laboratories
Eindhoven - The Netherlands

The paper describes a new approach to machine translation, based on Montague grammar, and an experimental translation system, Rosetta, designed according to this approach. It is a multi-lingual system which uses 'logical derivation trees' as intermediate expressions.

1. INTRODUCTION

Usually two approaches to machine translation are distinguished: the interlingual approach and the transfer approach (cf. Hutchins [1]). In the interlingual approach translation is a two-stage process: from source language to interlingua and from interlingua to target language. In the transfer approach there are three stages: source language analysis, transfer and target language generation. The approach advanced in this paper is a variant of the interlingual one. It requires that 'logically isomorphic grammars' are written for the languages under consideration. The syntactic rules of these grammars must correspond with logical operations, in accordance with the compositionality principle of Montague grammar. Moreover, the grammars must be attuned to each other as follows: if one grammar contains a rule corresponding with a particular logical operation, the other grammars must contain rules corresponding with the same operation. Syntactically, these rules may differ considerably. If the grammars are attuned to each other in this way, 'logical derivation trees', representations of both the syntactical and the logical structure of sentences, can be used as intermediate expressions.

The paper is organized as follows. In section 2 the relevant concepts of Montague grammar and the notion 'logically isomorphic grammars' are introduced. In section 3 a version of Montague grammar is described, called M-grammar, which is more suitable for computational use than Montague's original proposals. The property of logical isomorphy is then defined for M-grammars. In section 4 the design of the Rosetta translation system, based on this approach, is outlined, followed by a brief discussion in section 5.

2. LOGICALLY ISOMORPHIC MONTAGUE GRAMMARS

I will first introduce a few concepts of Montague grammar (cf. [2], [3]), in an informal way.

A Montague grammar defines a language by specifying (i) a set of basic expressions and their syntactic categories, (ii) a set of syntactic rules. Each rule specifies the categories of the expressions to which it is applicable, prescribes how these expressions must be combined to form a new expression and specifies the category of this expression. An expression is 'generated' by a Montague grammar if it can be derived by applying syntactic rules, starting from basic expressions.

Example 1. Assume that grammar G_1 defines a fragment of English.
Among G_1's basic expressions are 'Italian', of category ADJ, and 'girl', of category NOUN.
Among G_1's syntactic rules are:

R_4: if expression α is of category ADJ and β is of category NOUN, then $\alpha\beta$ is of category NOM.
R_7: if α is of category NOM, then 'the' α is of category NP.

One of the phrases G_1 generates is the NP 'the Italian girl'. It can be derived by applying R_4 to basic expressions 'Italian' and 'girl', and then applying R_7 to the result.

The way in which an expression is derived from basic expressions by application of rules can be represented by a tree, called derivation tree, with basic expressions labelling the terminal nodes and names of applied rules labelling the non-terminal nodes. For example, Figure 1 is the derivation tree of 'the Italian girl' according to grammar G_1.

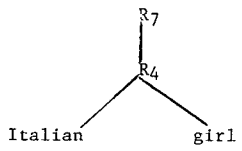

Figure 1

A Montague grammar must obey the compositionality principle, which reads: 'The meaning of a compound expression is composed from the meanings of its parts'. This is achieved by choosing the basic expressions and the rules in such a way that the meaning of a phrase can be defined by a syntax-directed translation in a logical language. For each basic expression a corresponding expression of the logical language is specified: the representation of its meaning. For each syntactic rule a logical composition rule is given, which shows how the meaning representation of the phrase constructed by the syntactic rule is derived from the meaning representations of the constituent phrases. The exact nature of the (intensional) logic that Montague uses, is not relevant here. For the present discussion it is important that a derivation tree displays not only the syntactic structure of a phrase, but its logical structure as well.

Example 2. Suppose grammar G_2 defines a fragment of Italian.
Among G_2's basic expressions are 'italiano', of category ADJ, and 'ragazza', of category NOUN.
Among G_2's syntactic rules are:
R_3': if expression α is of category ADJ and β is of category NOUN, then $\beta\alpha'$ is of category NOM, where α' is the adjective α adjusted to the number and gender of the noun β.
R_9': if α is of category NOM, then $\lambda\alpha$ is of category NP, where λ is a definite article in accordance with the number and gender of α.

G_2 generates the phrase 'la ragazza italiana', with the derivation tree of Figure 2.

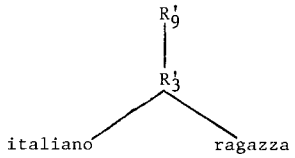

Figure 2

A comparison between the example grammars G_1 and G_2 shows that there is a correspondence between the basic expressions and rules of G_1 and those of G_2. 'Italian' and 'italiano' have the same meaning (in at least one of their readings) and the same holds for 'girl' and 'ragazza'. Rule R_4 and rule R_3' correspond with the same logical composition rule, the same holds for R_7 and R_9'. However, syntactically the rules differ considerably.

It may be possible to write grammars for large fragments of English and Italian and other languages in such a way that this semantic correspondence between basic expressions and rules of one language and those of the other languages is maintained. Dowty [4] has also pointed out this possibility (referring to similar observations by Curry and Dahl) and has given examples of correspondences between English, Japanese, Breton and Latin. I will call grammars that correspond with each other in this way logically isomorphic grammars. In the next section this notion will be defined precisely.

3. M-GRAMMARS

In Montague's original proposals [2] the syntactic rules operate on strings. From a linguistic point of view it is desirable to have rules operating on syntactic trees (cf. Partee [5]). From a computational point of view it is necessary to impose restrictions on the grammars in order to make effective parsing procedures possible. In an earlier paper [6] I developed a version of Montague grammar, called M-grammar, with the desired properties. I will briefly recapitulate the relevant definitions here. First, I will describe the kind of syntactic tree, called S-tree, on which the rules of an M-grammar operate.

An S-tree is a labelled ordered tree. The labels of the nodes may be compound entities of the kind more often met in computational linguistics, consisting of a syntactic category and a number of attribute-value pairs. The labels of the terminal nodes of an S-tree correspond with words [1]. The branches may be labelled with the names of syntactic relations (subject, head, modifier, etc.). The syntactic trees defined by a context-free grammar are a special, simple, kind of S-tree [2]. Each S-tree defines a phrase s, the sequence of terminal labels of t, called LEAVES(t).

An M-grammar defines a set of S-trees by specifying a set of basic S-trees (not necessarily terminal S-trees) and a set of rules, called M-rules.

An M-rule R_i defines a function F_i from tuples of S-trees to sets of S-trees. So application of R_i to tuple $t_1,\ldots,t_n$ results in a set $F_i(t_1,\ldots,t_n)$. If this set is empty, the rule is said to be not applicable. In order to make effective analysis procedures possible, M-rules must obey the following conditions.

Reversibility condition. Each rule R_i defines not only the compositional function F_i, but also an 'analytical' function F_i', from S-trees to sets of tuples of S-trees, in such a way that:

$$t \in F_i(t_1,\ldots,t_n) \iff \langle t_1,\ldots,t_n\rangle \in F_i'(t)$$

I will call F_i' the reverse of F_i.

Measure condition. There is a measure function μ, from S-trees to natural numbers, such that for each rule R_i the following holds:

$$\text{if } t \in F_i(t_1,\ldots,t_n) \text{ then } \mu(t) > \mu(t_j) \text{ for each } t_j.$$

So application of the analytical function F_i' results in a tuple of 'smaller' S-trees.

Analogously to section 2, we can define derivation trees, to be called D-trees here, which show the way in which an S-tree is derived from basic S-trees by application of M-rules.

For a given M-grammar two functions can be defined:
(i) M-GENERATOR, a function from D-trees to sets of S-trees. For each D-tree d M-GENERATOR(d) is the set of S-trees generated by applying the rules in d. M-GENERATOR is defined in terms of the compositional functions F_i.
(ii) M-PARSER, a function from S-trees to sets of D-trees. For each S-tree t M-PARSER(t) is the set of D-trees that generate t. M-PARSER is defined in terms of the analytical functions F_i'.
For both functions effective procedures can be written, thanks to the reversibility condition and the measure condition. It can be proved that for each D-tree d and S-tree t holds:

$$t \in \text{M-GENERATOR}(d) \iff d \in \text{M-PARSER}(t)$$ [3]

M-grammars must also obey the following condition.

Surface grammar condition. There must be a surface grammar G_S, such that the set of S-trees defined by the M-grammar is a subset of the set of S-trees defined by G_S. A surface grammar defines a set of S-trees, like an M-grammar, but with rules, called surface rules, which are simpler and less powerful than M-rules. If a surface rule is applied to a tuple of S-trees $t_1,\ldots,t_n$ it creates an S-tree with a new top and with $t_1,\ldots,t_n$ as immediate subtrees. A context-free grammar is a special case of a surface grammar.

For a surface grammar a parser can be defined: an effective function procedure, to be called S-PARSER, which assigns to any phrase s the set of grammatical

S-trees t such that LEAVES(t) = s.

Thanks to the surface grammar condition, M-PARSER can be extended to a function from sentences to D-trees. First the function S-PARSER, defined by the surface grammar, is applied to the sentence and then M-PARSER is applied to each of its results. M-GENERATOR can be extended to a function from D-trees to sentences by applying the function LEAVES to each S-tree in M-GENERATOR(d).

The basic expressions and the rules of an M-grammar must be chosen in accordance with the compositionality principle. For each basic expression b_i a set of logical expressions L_j must be specified, representing the meanings of b_i. For each M-rule R_I a logical composition rule S_j must be specified (several M-rules may share the same S_j).

Let us now define a logical D-tree as a D-tree with names of logical composition rules S_j at the non-terminal nodes and names of 'basic' logical expressions at the terminal nodes. For each M-grammar a function LOG from (syntactical) D-trees to sets of logical D-trees can be defined, as well as the reverse function LOG'. If the example grammars G_1 and G_2 were to be reformulated as M-grammars, Figure 3 might be a logical D-tree for the D-tree of Figure 1, and also for the D-tree of Figure 2.

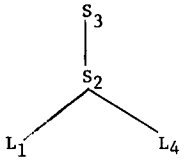

Figure 3

A logical D-tree defines exactly one logical expression and is in fact a redundant intermediate step between syntactical D-tree and logical expression. However, as we will see in the next section, it is logical D-trees and not logical expressions which act as the pivot of the translation process in the Rosetta system.

We are now able to define for each M-grammar the function ANALYSIS, which maps sentences to sets of logical derivation trees and the reverse function GENERATION.

$$\text{ANALYSIS}(s) =_{\text{def}} \{e \mid \exists t, \exists d: t \in \text{S-PARSER}(s) \wedge d \in \text{M-PARSER}(t) \wedge e \in \text{LOG}(d)\}$$

$$\text{GENERATION}(e) =_{\text{def}} \{s \mid \exists t, \exists d: d \in \text{LOG}'(e) \wedge t \in \text{M-GENERATOR}(d) \wedge s \in \text{LEAVES}(t)\}$$

(s ranges over sentences, t over S-trees, d over D-trees, e over logical D-trees)

The following theorem can be proved easily.

Reversibility Theorem.

$$\forall s, \forall e: \; e \in \text{ANALYSIS}(s) \iff s \in \text{GENERATION}(e)$$

The given definitions enable the notion of logical isomorphy to be defined precisely. Let us assume as given two M-grammars G_i and G_j with generation functions GENERATION_i and GENERATION_j.

$G_i \sim G_j$ (G_i logically isomorphic with G_j) iff

$$\forall e: [\; \exists s: s \in \text{GENERATION}_i(e) \iff \exists s': s' \in \text{GENERATION}_j(e) \;]$$

(for each logical D-tree assigned to a sentence s by G_i, there is a sentence s' to which G_j assigns the same logical D-tree, and vice-versa)

Proving that two grammars are logically isomorphic may be complicated. It is simple for grammars of which the rules are complete (i.e. applicable to all expressions of the required categories), if there is a one-to-one correspondence between the syntactic categories, the basic expressions and the rules of the two grammars.

Because the relation $\sim$ is an equivalence relation, it makes sense to speak of a set of logically isomorphic grammars, as I will do in the next section.

4. THE ROSETTA SYSTEM

Suppose we have logically isomorphic M-grammars G_1, G_2,... for languages L_1, L_2,.... In section 3 we have seen that each grammar G_i defines a function ANALYSIS_i and a function GENERATION_i. These functions determine a translation function for each pair of languages L_i, L_j.

$$\text{TRANS}_{ij}(s) =_{def} \{ s' \mid \exists e : e \in \text{ANALYSIS}_i(s) \wedge s' \in \text{GENERATION}_j(e) \}$$

(s and s' are sentences, e is a logical D-tree)

An experimental translation system, Rosetta, has been designed, which translates isolated sentences of a source language into sets of sentences of a target language, according to the definition of TRANS_{ij}. Given isomorphic grammars for a set of languages the source language and the target language can be freely chosen from this set. The definition of logical isomorphy and the Reversibility Theorem guarantee that - given correct grammars - each sentence of any source language will be correctly translated into any target language, for each meaning of that sentence.

In Figure 4 the various stages of the translation process in Rosetta are shown. The example expressions are simplified, e.g. the attribute-values have been omitted in the S-trees. MORPH and MORPH' are the components that perform dictionary look-up and morphological rules during analysis and generation.

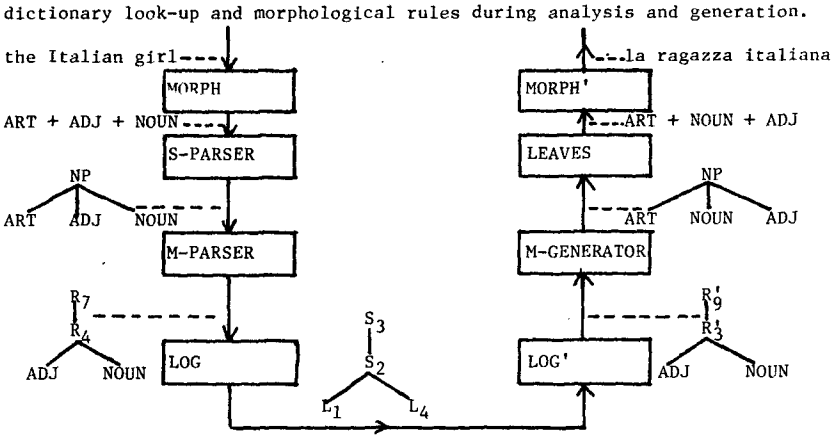

Figure 4

It should be noted that LOG and LOG' may perform more complex operations than Figure 4's example suggests. Basic expressions may be larger units than words. Furthermore an adjective of the source language may be translated into a prepositional phrase or a relative clause of the target language, because these categories correspond with the same logical type as the category adjective. Rule S_2 is not only translated into R'_3, but also into rules that combine a noun with a prepositional phrase or a relative clause. Which of the combinations of these local ambiguities is correct is decided in M-GENERATOR.

The first version of Rosetta was designed and implemented in 1981, for very small fragments of Dutch, English and Italian. The system operates in a breadth-first manner: at each level it generates all results, in the case of ambiguities, and ultimately it gives all possible translations. The program was written in PASCAL and runs under UNIX on a VAX 11/780.

There are relatively few applications for a system delivering all syntactically possible translations. As is well-known, in order to choose the 'best' out of the possible translations, knowledge about the world and the context is needed. A future version of Rosetta will presumably be provided with interactive facilities that enable the user to contribute this kind of knowledge, as in ITS [7].

5. DISCUSSION

There is a more obvious way to base a multi-lingual translation system on Montague grammar, or on logic in general, than the one described here: use the logic itself as the interlingua. The ϵ-λ-context-free language of the SALAT system [8] is an example of this. Friedman [9] reports on work by Godden, who uses Montague's Intensional Logic.[4] The success of the Rosetta approach depends on the correctness of the hypothesis that logically isomorphic grammars can be written for interesting fragments of languages. At first sight it may seem that we can avoid the necessity to attune the grammars to each other if we do not use logical derivation trees but the logical expressions themselves as intermediate expressions. However, the important distinction is not between using logical derivation trees and using logical expressions, but between making use of the form of logical expressions and not doing so. In a powerful logical language each meaning can be represented by an infinite variety of logical forms. If a generation component has to be able to generate a sentence not only from a particular logical form, but also from all logically equivalent forms (which might be the result of the analysis component of some other language), this will cause decidability problems. These are in practice avoided by severely limiting the possible forms that analysis components may produce and ensuring that the generation components can handle these forms (The same holds for systems that use syntactic deep structures as intermediate expressions). The point I want to make here is that this is just another - less explicit - way to attune the grammars of the languages to each other.

The reversibility of the M-grammars used in Rosetta has the advantage that the same grammar can be used for analysis and generation.[5] Futhermore it makes testing of the system easier: if the analysis component and the generation component of the same language are coupled, each sentence must be one of its own possible translations. Ultimately, for most applications it will be necessary to give up this nice symmetry, to make the analysis more tolerant and the generation more restrictive. But such modifications should be the result of conscious decisions and not be mixed up too soon with the incapability to write correct and complete grammars.

NOTES

1) The relation between the words and the complex labels of the terminal nodes has to be defined by a dictionary in combination with morphological rules. This component is not discussed here.

2) In [6] the restricted - context-free - definition of S-tree is used.

3) For grammars with syntactic variables, the symmetry between M-GENERATOR and M-PARSER holds only for 'canonical' D-trees, in which the indices of syntactic variables are chosen in a restricted way. This can be done without loss of generality. Cf. [6].

4) Another translation system based on Montague grammar is described by Nishida et al [10]. But here the logic is not used as an interlingua, but as the level where the transfer, from English to Japanese, takes place.

5) In the current implementation of Rosetta the rules are not automatically compiled or interpreted from the original notation. The analytical and generative versions of the rules are 'hand-compiled' into PASCAL.

ACKNOWLEDGEMENT

The Rosetta system was designed and implemented in cooperation with Joep Rous.

REFERENCES

[1] Hutchins, W.J., Machine translation and machine-aided translation. Journal of Documentation, Vol. 34, No. 2 (1978) 119-159.

[2] Thomason, R.H. (ed.), Formal Philosophy, Selected papers of Richard Montague (Yale University Press, New Haven, 1974).

[3] Dowty, D.R., Introduction to Montague semantics (Reidel, Dordrecht, 1981).

[4] Dowty, D.R., Grammatical relations and Montague grammar, to appear in: Pullum,G. and Jakobson, P. (eds.), On the Nature of Syntactic Representation.

[5] Partee, B.H., Some transformational extensions of Montague grammar, in: Partee, B.H. (ed.), Montague Grammar (Academic Press, New York,1976).

[6] Landsbergen, J., Adaptation of Montague grammar to the requirements of parsing, in: Groenendijk, J.A.G., Janssen, T.M.V. and Stokhof, M.B.J., Formal Methods in the Study of Language Part 2 (MC Tract 136, Mathematical Centre, Amsterdam, 1981).

[7] Melby, A.K., Smith, M.R. and Peterson, J., ITS: Interactive Translation System (Proceedings COLING 80, Tokyo, 1980).

[8] Friedman, J., Expressing logical formulas in natural language, in: Groenendijk, J.A.G., Janssen, T.M.V. and Stokhof, M.B.J., Formal Methods in the Study of Language Part 1 (MC Tract 135, Mathematical Centre, Amsterdam, 1981).

[9] Hauenschild, C., Huckert, E. and Maier, R., SALAT: machine translation via semantic representation, in: Bäuerle, R., Egli, U. and Stechow, A. von (eds.), Semantics from Different Points of View (Springer Verlag, Berlin, 1979).

[10] Nishida, T., Kiyono, M. and Doshita, S., An English-Japanese machine translation system based on formal semantics of natural languages (Technical Report, Tokyo Univ., 1981).