[Proceedings of the *Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Colgate University, Hamilton, New York, August 14-16, 1985]

LMT: A PROLOG-Based Machine Translation System

Michael C. McCord

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Extended Abstract

The talk will describe a machine translation system, **LMT**, based in PROLOG, translating from English to German. The effort on **LMT** *per se* has just begun this year, although the logic programming methodology for the analysis of the source (English) goes back several years (see, *e.g.,* McCord, 1982).

Analysis of the source is essentially independent of the task of translation, and in fact the very same English grammar used in **LMT** is also used in an English data base query system. Analysis is based on a logic grammar formalism, *Modular Logic Grammars* (MLG's) (McCord, 1985). MLG's have clearly separated syntactic, lexical, and semantic components. Furthermore, syntactic structure building is handled automatically by the syntax rule compiler, making grammars more compact (than, say, Definite Clause Grammars) and easier to understand. Syntax rules are compiled directly into PROLOG, so that parsing is top-down.

The interface of syntax to the (source) lexicon promotes modularity in two ways: (1) Syntactic analysis of verb and noun complements is driven by slot-filling with slot-frames obtained from lexical entries, so that a great deal of the "power of parsing" is given to the lexicon. (This is similar in spirit to the Lexical Functional Grammar of Kaplan and Bresnan (Bresnan, 1982)), though developed independently.) The slots used are *syntactic* slots (like 'subject', 'object', and 'indirect object'). (2) Semantic categories are not mentioned explicitly in the syntax rules, although they get used during parsing through semantic types appearing in the slot frames for explicit lexical entries. Semantic constraints on the parse are exercised by PROLOG unification of semantic types. These types are trees that allow cross-classification.

Because of this clean interface to the lexicon, the syntactic component is truly syntactic (hence more generally applicable and easier to understand), yet parse trees actually show the results of semantic choices. Terminal nodes contain (disambiguated) word senses with their complements indicated as logical-variable arguments. And of course phrase attachments are constrained through the use of semantic type matching. In overall shape, these analysis trees, built automatically by the syntax rule compiler, are like surface-structure trees, but they differ from

derivation trees in certain ways that will be described in the talk. These analysis trees are used as the point of transfer in **LMT**.

The semantic component for an MLG takes parse trees and produces logical forms in a higher-order logic, **LFL**. In producing these logical forms, decisions must be made about scoping of quantifiers and other modifiers such as adverbs and adjectives. **LFL** analyses are used in the data base query system. Originally, **LFL** analyses were intended as the point of transfer for the English-German translation system. Reasoning behind the decision to use the parse trees will be given in the talk.

Given an MLG parse tree, the translation system works basically in three phases: Transfer, Restructuring, and Inflection.

The Transfer phase produces a tree of the same shape as the English parse tree, but with the following differences in the node labels. Most terminal nodes are labeled by German words (translations of the corresponding English word senses) in uninflected form together with feature structures that will control inflection. Some terminal nodes are labeled by special symbols that trigger transformations in the Restructuring phase. Non-terminal nodes are labeled by structured phrase category symbols appropriate for German phrases.

A fairly good use of PROLOG techniques is made in the Transfer phase. In the transfer of lexical items, unification and logical variables are used in two ways. (1) The English word-sense predications appearing in the English parse tree actually have *semantically typed* variables as their arguments. The transfer dictionary entries require particular semantic type configurations on the English word-sense predications, for the purpose of German word selection, and these type configurations are required to unify with the actual type configurations appearing with the English word sense (which themselves were created during parsing by unification). (2) The variables appearing in the English word-sense predications are like pointers to the complements of the word, and so these variables can be used to exercise case requirements on the corresponding German complements without doing any searching for them in the tree. The transfer lexical entry unifies each such "pointer" variable with the case required by the German word for that complement. A *case* could be a normal case like 'dative', but could also name a specific preposition together with the (normal) case for its object noun phrase.

The English parse trees are rather dependency-oriented (where each phrase has a head word), and correspondingly the processing in Transfer is head-oriented. (In each cycle of the recursion, the head and the phrase label of a phrase are translated first, then all the modifiers.) Logical variables are used to advantage here, because the feature structure for the head contains logical variables that get bound only after processing the modifiers.

Care is taken of course in Transfer to make sure that feature agreement reflects the *German* requirements, not just the original English. This means paying attention to the situations, for instance, where the subject of the German verb does not correspond to the English subject and where a German noun must have a different number from that of the original English.

The Restructuring phase actually has two passes. In the first, tree transformations are applied, recursively on every level. The pattern-matching for these transformations is mainly PROLOG unification, but there is an augmentation that allows one to use pattern variables that match sublists. Transformations can of course add, delete, and move items. An example of a transformation that adds is a Relative Clause transformation, which adds the German relative pronoun in a relative clause. The English analysis tree shows no relative pronoun as a tree node (recall that relative pronouns can be implicit in English), although the relative clause node label contains a logical variable corresponding to the fronted item (unified with the appropriate argument of the verb head of the relative clause). The Relative Clause transformation looks at this "trace" variable in creating the relative pronoun, and the proper case is automatically associated with the German relative pronoun because of the transfer processing of the verb head of the relative clause (which contains the trace variable). A transformation that deletes and moves is one that handles yes-no sentences having a "do" auxiliary verb, where the "do" is replaced by the translation of the main verb.

The second pass in Restructuring is a simple reordering (sorting) of items on every level. This includes the movement of the verb toward the end of a dependent clause. But such verbs are not moved *too far* to the right, pass, say, complement clauses and relative clauses. Such "blocking" clauses are themselves raised out of lower nodes by a transformation (operating cyclically) in the first pass of Restructuring, so that their presence is easy to detect in the reordering pass.

Inflection occupies a separate pass because the information residing in the feature structure components of terminal nodes is not complete until the previous passes (mainly Transfer) have operated completely. Originally, these feature structures contain logical variables that get bound only after processing of modifiers in Transfer. Inflection itself is quite straightforward, just using the citation forms of words and the associated feature-structure terms against tables of inflections (and exceptions) stored as PROLOG unit clauses.

The implementation is in VM/PROLOG (from the IBM Paris Scientific Centre) on an IBM 3081. Processing seems efficient so far. For example, the following translation:

The old man knew that the car he had bought belonged to somebody
he knew or had seen.

Der alte Mann wußte, daß der Wagen, den er gekauft hatte, jemandem gehörte,
den er kannte oder gesehen hatte.

was done in 48 milliseconds (including the analysis of the English).

References

Bresnan, J., editor. *The Mental Representation of Grammatical Relations,* MIT Press, Cambridge, Massachusetts.

McCord, M. C. (1982) "Using slots and modifiers in logic grammars for natural language," *Artificial Intelligence,* vol 18, pp. 327-367. (Appeared first as 1980 Technical Report, University of Kentucky.)

McCord, M. C. (1985) "Modular Logic Grammars," *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics,* pp. 104-117.