[From: Paul L.Garvin (ed.) *Natural Language and the Computer* (New York: McGraw Hill, 1963)]

PAUL L. GARVIN

Syntax in machine translation

The general problem area of machine translation has been outlined and related to other fields of language-data processing in Garvin's "A Linguist's View of Language-data Processing." It has been discussed in some detail in the two immediately preceding sections by Hays and Harper. This discussion will deal more specifically with the question of syntax: both with the place of syntax in the over-all design of a machine-translation program, and with the detailed problems of syntactic resolution.

Machine translation is a process for translating text automatically from one language to another. This process is embodied in a program—that is, a set of instructions for a computer. With such a program, any large-scale computing machine can be transformed into a translating machine.

A genuine translation is more than just the replacement of individual foreign words by individual English words. It must transmit the essential information contained in the foreign text to an English reader who is unfamiliar with the language. The meaning of each sentence is more than the sum of the meanings of each word—it also depends on the structure of the sentence and the function of each word in that structure. To obtain a satisfactory translation, the program must therefore contain not only a dictionary lookup for finding individual word meanings, but also a translation algorithm for making correct translation choices and for detecting the elements of meanings implicit in the sentence. The core of such a translation algorithm is a *syntax routine.*

The major purpose of a syntax routine in machine translation is to recognize and appropriately record the boundaries and functions of the various components of the sentence. This syntactic information is essential for the efficient solution of the problem of word order for the output and is equally indispensable for the proper recognition of the determiners for multiple-meaning choices.

It is furthermore becoming increasingly apparent that it is the design of the syntax routine which governs the over-all layout of a good machine-

translation program and lends it the unity without which it would remain a patchwork of individual subroutines and piecemeal instructions. The conception of syntax thus becomes important beyond the immediate objectives which the routine serves in the program.

A syntax routine for machine translation can be described from two viewpoints: (1) the underlying linguistic conception of syntax (the syntactic "model"); and (2) the logical technique employed in the design of a syntax program.

Two kinds of linguistic conceptualization underlie current work in machine translation: formal or quasi-formal models of language on the one hand, and more discursive conceptual frameworks on the other. The former are illustrated by the transformational model and the dependency model, as discussed earlier by Stockwell and D. G. Hays respectively. The latter are illustrated by the informational and definitional models, as presented by Sebeok and Garvin.

Formal models attempt to base a conceptualization of language upon a previously derived system. In machine translation, D. G. Hays and his coworkers have extensively applied an approach based on the dependency model referred to above. Discursive frameworks, on the other hand, can be developed by a formalization of common-sense criteria and the systematization of the experience of trained observers. In Russian-English machine translation, such common-sense approaches largely consist in a modification of traditional Russian grammar for data-processing purposes. Since the traditional grammar of Russian is, on the whole, not too far removed from linguistic reality (see "The Definitional Model of Language"), the various adaptations to machine translation undertaken by American research groups have proved quite adequate.

The conception of linguistic structure as presented here, in so far as it concerns syntax, is comparable to what has become known as the *immediate-constituent model*, but with some significant differences. Where the immediate-constituent approach takes the maximum unit—the sentence-as its point of departure and considers its step-by-step breakdown into components of an increasingly lower order of complexity, we can start out with the minimum unit—the morpheme in straight linguistic analysis, the typographical word in language-data processing—and consider its gradual fusion into units of increasingly higher orders of complexity, which we have termed *fused units*. A sentence is thus viewed not as a simple succession of linear components but as a compound chain of fused units of different orders of complexity variously encapsulated in each other. Syntactic analysis, including the automatic analysis which an MT syntax routine must perform, then has as its objective the delineation of this encapsulation of fused units by ascertaining their boundaries and functions.

For MT purposes, the most significant order of fused units are the major members—subjects, predicates, and objects—of clauses. The lan-

guages of the world differ significantly in the manner by which these clause members are formally marked. In a language such as Russian, for instance, the clause members are characterized primarily by a particular selection of grammatical endings, the functions of which must often be ascertained from the context. Only in case of unresolvable ambiguity of endings will conditions of word order and the semantic nature of particular words indicate the functions of the clause members. In languages such as English or French, on the other hand, word order is the primary means for identifying and characterizing the clause members.

A recognition routine for Russian syntax must therefore in some way search the span of the entire clause to identify appropriate endings, and from them ascertain the boundaries and functions of the major clause members. The corresponding command routine for translation into English or French must then have the capability for the rearrangement of entire clauses, in order to generate correct word order for the output language.

A recognition routine for English or French syntax, on the other hand, can draw on word order in the identification of the major clause members. This identification, however, presupposes that the grammatical function of each constituent word of the clause has been unambiguously ascertained. Since in English, for instance, a great many words have ambiguous part-of-speech membership (for example, *love* functioning as a verb, noun, or modifier, depending on the context), the identification of the clause members can not proceed until these ambiguities have been resolved. A command routine for translation into Russian must then generate correct grammatical endings and stem modifications where necessary, in order to render the functions of the major clause members which in English or French are marked by word order.

The following discussion will be limited to a detailed consideration of the problems arising in the machine translation of Russian into English. The fused-unit conception is particularly well suited to MT since the minimum units—for this purpose the typographical words—constitute the primarily given input units; the program then computes the fused units and their interrelations from the grammar codes (see below) of the words. The methodological basis used here for this computation is what we can call the *fulcrum approach* to syntax. Its basic features have been adopted and have been implemented to a considerable extent by the machine translation group at Thompson Ramo Wooldridge Inc. This approach is a suitable point of departure for the discussion, not only of our own work in syntactic resolution, but also of the work of other groups that use a common-sense grammatical framework as the basis for their approach to Russian-English MT.

The fulcrum approach consists in directing the primary syntactic searches toward those pivot words (fulcra) within the sentence, around

which other words are centered. The fulcra contain the maximum amount of grammatical information, thereby allowing an optimization of further searches within the sentence. Not only is the sentence as a whole assumed to have a fulcrum, but the various constituents of a sentence in turn are assumed to have their fulcra.

Thus, the fulcrum of the Russian main clause is assumed to be its predicate. Once the predicate is known, it allows a reasonable prediction concerning possible subjects and objects. The number (and, when present, the gender) of the predicate will be in agreement with the subject and hence will predict the corresponding features of the latter. The predicate has certain inherent government characteristics: some verbs govern the accusative, other verbs govern the instrumental, etc. These government characteristics of the predicate allow prediction of the corresponding characteristics of the object(s). The converse does not apply.

The fulcrum of a nominal block (that is, a noun with modifiers and other dependents) is assumed to be the noun: It is that member of the block which determines the case and number forms of the other members, and not conversely.

The syntactic searches, whether based on an overt fulcrum approach or not, are made possible by including, with each word of the machine dictionary, a grammar code containing all the grammatical information inherent in the particular word. The grammar code is thus a representation of all the potential functions and relations of a word; the syntactic searches then serve to determine which of these inherent possibilities actually does apply for the particular sentence in which the word occurs.

The grammar code utilized by TRW is organized in terms of this potential function of the words, rather than simply their morphological shape, particularly for the indication of parts of speech in the grammar code. In traditional Russian grammar, parts of speech are largely determined on the basis of the nature of the stem and the grammatical endings. In our grammar code, priority is given to the capability of a word to constitute the predicate, subject or object, or other functional component of the sentence. Increasing emphasis on functional rather than morphological grammar coding is becoming apparent in the work of other groups as well.

Thus, words with the same functional capabilities are given the same code designation, even though in traditional Russian grammar they are not considered the same part of speech. For example, some Russian predicative adjectives function as predicates of sentences in much the same way as verbs (barring certain ambiguities to be discussed further below). They, together with verbs, are given the same grammar code digit for "predicativeness." Another example are certain modifying adjectives which share an important characteristic in common with participles—both may be accompanied by certain governed structures (for example,

"analogous *to this,*" "pertaining *to this").* These are jointly coded as *governing modifiers.*

On the other hand, words which in traditional grammar are considered different forms of the same part of speech are treated separately in our grammar code if they have different functional potential. An example of this are infinitives, which are traditionally considered merely a form of the verb. Since, however, they differ significantly from finite verb forms (e.g., by not having a capability for taking a subject), our grammar code assigns to them a separate "infinitive" digit, while finite verb forms are coded for "predicativeness."

The actual syntax routines utilize the information contained in the grammar code for a series of searches intended to detect and record the interrelations of the words of each sentence and the function of the words within this net of relations. Two major methods for organizing these syntactic searches deserve mention. The first is a search pattern in which all syntactic information is immediately recorded from the grammar code as each word is encountered; the second consists of a sequence of passes at the sentence, in which each pass is designed to retrieve and record information about a particular set of word relations and functions, in order to identify fused units of a particular order and type. Further information is retrieved in later passes. A well-known and very ingenious example of the first-mentioned search pattern is the method devised by Mrs. Ida Rhodes of the National Bureau of Standards, and named *predictive analysis* by her followers in the machine-translation group at the Harvard Computation Laboratory. The TRW group, on the other hand, uses a pass method.

A question of logical flow, less related to linguistic considerations than the above, is the extent to which the syntactic algorithm is based on a table-lookup or a logical-tree principle. A table lookup, as the term indicates, consists in the storage of information in tables in memory, to be looked up when a particular subroutine calls for it. A logical tree, on the other hand, in this connection refers to a program design in which a small amount of information is looked up at a time, and the subroutines of the program branch extensively in terms of the answers obtained to the questions asked by the program.

The argument in favor of a table-lookup approach to syntactic programming is that the same small lookup routine can be used to call a variety of tables and, hence, yields a more flexible program; and that even large tables are easier to construct and revise than complex logical trees with many branches. Logical trees by contrast are favored because of their greater relative efficiency, and because of the concern that tables, even if well organized, become unwieldy once they increase beyond a certain size. The TRW syntax algorithm is largely designed as a logical tree and uses comparatively few tables.

The advantage of a pass method implemented by a logical-tree design is that instead of having to account, at each step of the left-to-right search, for each of the many possibilities contained in the tables, every pass is limited to a particular search. With the proper sequencing of passes, the syntactic retrieval problems presented by each sentence can be solved in the order of their magnitude, rather than in the accidental order of their appearance in the text.

In a program based on the pass method, each individual pass is laid out in terms of the information available when the pass is initiated and in terms of the objective that the pass is intended to accomplish. These two factors are closely related to each other, in that the output of a preceding pass becomes the input of the subsequent pass. The scope of each pass and the order of the various passes thus together present the most significant design problem of the program.

The various passes are concerned either with individual words and strings that function as single words (such as symbols and numerals), or with fused units that the program labels as *word packages*, in order to treat them as single entities. The former set of passes serves to ascertain the function of a particular word or string in order to assign to it the grammar code required for further syntactic processing in the passes designed for word packaging. The purpose of the latter is to identify the boundaries and functions of the word packages that constitute the components of the sentence. The word-packaging passes first identify the potential fulcrum of a given package and then use it as the initial point from which to search for the required boundary and function information to delimit and define the package.

The linguistic considerations affecting the design of a syntax program stem from the assumption that the various orders of units and their relations do not have the same degree of significance for the over-all structure of the sentence. An adequate analysis of the sentence must give priority to the identification of the major sentence components (subjects, predicates, objects), since the relations between these components are the focal point around which the remaining syntactic relations are centered.

Applying this to the organization of the passes, it means that the main syntax pass—that is, the pass designed to identify the boundaries and functions of the major clause members of the main clause—becomes the pivot of the program. The remaining passes can be laid out in terms of the input requirements and expected output of this central pass. Preceding it will be preliminary passes designed to assign grammar codes to words which are not in the dictionary (a missing-word routine) and to aberrant typographical matter such as symbols and formulae, as well as passes designed to compute from the grammar codes information needed as input to the main syntax pass. Following it will be terminal passes,

the function of which is to fill the gaps in syntactic information remaining after the main syntax has accomplished its objective.

The present TRW syntax algorithm works in the following manner: To find a fulcrum, the program reads the word-class field of the grammar code of each word that the lookup has brought into the work space. This may be either an original grammar code as brought in from the dictionary, or a revised grammar code as assigned to a word or string by an earlier pass. If the word is of a class that may function as the fulcrum of a word package, this information serves as the signal for later calling the subroutine designed to identify the boundaries and possible function of the package in question.

Each pass of the program is concerned with either the assignment of an appropriate grammar code or with the identification of an appropriate word package. The word packages correspond more or less closely to the fused units of the Russian structure; the order of passes constitutes the sequence in which the search for the various fulcra and fused units is conducted, with the aim of the correct recognition of their encapsula on.

The syntax program consists of four series of passes:

Preliminary passes are designed to insure that all the words and strings of the sentence are provided with the appropriate unambiguous grammar codes.

The preliminary passes are required by the discrepancy between the information contained in the grammar code and the information necessary for the main syntax passes. The grammar code furnishes three sets of indications: word-class membership, agreement characteristics, and government characteristics. As is well known, for each dictionary entry some of this information will be unambiguous, some ambiguous, depending on the particular word forms involved.

Aside from accidental typographical homonyms (such as *est'* meaning *is* or *to eat),* grammatical ambiguities relate to word-class membership and agreement characteristics (where ambiguities as to government characteristics are found, they are dependent on another grammatical function, that of word-class membership).

While the main syntax passes may tolerate agreement ambiguities, they cannot admit word-class ambiguities in their input, since the fulcrum approach is based on the recognition of the fulcra by their word-class membership. One of the essential functions of the preliminary passes is thus the resolution of ambiguous word-class membership.

It is furthermore reasonable to expect that sentences will contain discontinuous fused units—that is, fused units interrupted by variously structured intervening elements. Unless such intervening structures are properly identified in prior passes, the program will not be able to skip

over them in the search for elements functionally relevant to the objectives of the later syntactic passes.

Finally, since the internal structure and external functioning of units are relatively independent of each other (as discussed in "The Definitional Model of Language"), a number of constructions can be expected within each sentence which by their internal structure resemble potential major clause members, but do not have that external functioning.

An example of this are relative clauses: Their internal structure resembles that of a main clause, and they contain similarly structured clause members, but their external functioning is that of inclusion in nominal blocks as modifying elements. Constructions such as these have to be identified by appropriate prior passes, and their boundaries and functions recorded for inclusion in the main syntax.

Minor syntax passes are designed to identify and label the word packages which are candidates for inclusion in, or exclusion from, the major sentence portions upon which the next series of passes operates.

Major syntax passes are designed to identify and label the major portions of the sentence.

Terminal passes are designed to identify and label certain word packages not previously identified and labeled.

The following passes enter into each of the four series in the present TRW program:

The *preliminary passes* include a numeral-and-symbol pass and a set of homograph-resolution passes. The numeral-and-symbol pass serves to assign a grammar code to number-symbol strings in the sentence, the homograph-resolution passes serve to resolve various systematic word-class ambiguities (such as the well-known predicative adjectives ending in *-o,* which also function as adverbs) .

The *minor syntax passes* include a nominal-blocking pass, a prepositional-blocking pass, an inserted-structure pass, and a governing-modifier pass.

The purpose of the nominal-blocking and prepositional-blocking passes is to identify and label nominal blocks and prepositional blocks respectively. A nominal block consists of a noun and its accompanying modifiers; a prepositional block consists of a preposition and the nominal block which it governs.

An inserted structure is one which is not grammatically related to the remainder of the sentence (comparable to English *as it were* in a sentence such as *this, as it were, can be considered an inserted structure*). The purpose of the inserted-structure routine is to identify and label these structures so that they can be skipped over by later syntactic searches.

The purpose of the governing-modifier pass is to identify and label governing-modifier packages. The word class of governing modifiers in-

eludes both attributive participles and those adjectives which can in turn govern certain dependent structures (see above).

A governing-modifier package is the word package which contains a governing modifier together with the structures that it governs. The fulcrum of this package is the governing modifier; the program reads its word-class code to call the appropriate routine, then reads the government code in order to search for an appropriate governed structure.

The *major syntax passes* include a clause-boundary-determination routine, a relative-package routine, and the main syntax routine.

The clause-boundary-determination routine serves to ascertain the boundaries between the several component clauses of a compound sentence, in order to process one clause at a time.

The relative-package routine identifies relative clauses by finding the relative pronouns introducing them. It then sets the clause boundary to allow the main syntax routine to process the components of the relative clause. Finally, the package as a whole is labeled for inclusion in the appropriate nominal block.

The main syntax routine serves to identify the major clause components—namely, subject, predicate, and object—and to ascertain their boundaries. The routine first searches for the fulcrum of the clause, the predicate, and then uses the information derived from the grammar code of the predicate to search for the subject and object.

Two *terminal* passes are included in the present program: *(a)* a predicate-packaging pass, serving to include adverbs and other dependent words together with the predicate in a predicate package; and *(b)* a genitive-blocking pass, serving to identify genitive nominal blocks and attach them to the preceding nouns which are likely to govern them. Both passes identify and label the packages, and set their boundaries.

A syntax program of the kind described above will produce a record of the syntactic structure of the sentences of the input text, as recognized by the program. This syntax record furnishes a significant part of the information required for the correct application of the command routines for word rearrangement and multiple-meaning choice. At the present developmental stage of machine translation, an important function of the syntax record is its use in checking out the operation of the syntax program. It allows the analyst to determine what routines have been applied and why, and to introduce required modifications on the basis of an examination of the translation output and the syntax record.

At Thompson Ramo Wooldridge Inc., the syntax record is stored on an information tape which can be printed out separately. It contains the grammar codes of each word (whether brought in from the dictionary or assigned by the program), as well as an indication of the word packages that have been identified and additional syntactic and semantic information.

The remaining information required by the command routines is semantic. It has to be retrieved by a semantic program linked to the syntax. The semantic program can deal with the meanings of the input words on the basis of an appropriate semantic code contained in the machine dictionary. The code will serve to call the routines needed for the resolution of semantic choices.

Semantic resolution presents a much more difficult problem area than syntax. Our knowledge of the inherent semantic system of a language is as yet sketchy and indefinite (see "A Linguist's View of Language-data Processing"). Consequently, the semantic codes used in present machine-translation programs are not as systematic as the grammar codes. The resolution routines are piecemeal, and what coordination there is in these routines is based on their link to the syntax program. An organizing principle for semantic information, comparable to the fulcrum approach to syntax, will have to be found before a more efficient and exhaustive approach to semantic resolution can be envisioned.