[From:*Translation: Applications and Research*, ed. Richard W. Brislin (New York: Gardner Press, 1976), pp.247-259]

An Operational Machine Translation System

PETER TOMA

ALL HUMAN LANGUAGES follow rules. Any precisely formulated rule can be programmed. An entire set of programmed rules comprises a computer translation system. The feasibility of computer translation is determined by the extent to which the rules of human language can be defined.

I.

Translation is one of the most fascinating—and complicated—intellectual tasks man performs. In fact, Richards (1953) has gone so far as to describe this type of communication process as "what may very probably be the most complex type of event yet produced in the evolution of the Cosmos." Perhaps it is this complexity that lends such fascination to the job of automatizing this human process on today's fastest and largest computers.

When humans translate, they first read the text in its original (source) language. To do this, they need to know

1. What the individual words mean.
2. the role each word plays in any particular sentence.
3. whether the denotation of any word(s) is affected by its context.

After the translator has determined these, he transfers the information content into the target language; that is, he produces the translated text. When the translator is human, it is

virtually impossible to produce a translation free from the translator's *interpretation* of the original textual information. Necessarily, an interpretation is tinged by the human translator's ability to grasp the concept expressed in the source language, and, depending on the type of material to be translated, his own reactions to that concept.

Machine translators or computer translation systems need to know the same things, basically, that human translators do. Therefore, they must have adequate dictionaries ("vocabularied"), syntactic analysis programs, semantic analysis procedures, and target language synthesis programs.

There have been many publications devoted to computer translation and many discussions of the problems that arise in developing, or attempting to develop, computer translation systems. The first, Locke and Booth's *Machine Translation of Languages* appeared in 1955. Yet the first demonstration of a real translation system did not occur until June 6,1959, when, at the Pentagon, the Georgetown (SERNA) system (described in Toma, 1959) translated over a hundred thousand words of it had never "seen" before.

The SERNA system ran on second-generation computers, whose input-output operations as well as core processing operations were so slow that it seemed impossible that computer translation would ever be economically feasible.

Work with the SERNA system, and with other systems designed during the ensuing years, made it apparent that it would be necessary to know a great deal more about human languages and the rules we follow to use them than anyone knew at that time.

In optimizing and refining SYSTRAN, the operational system described below, we find that this is still true. In too many instances there are no grammar books or linguistic analyses to turn to for answers. Rather, programs are written to have the machine produce appropriate corpora for linguists to analyze. The results of the linguists' work are then programmed into the translation system.

SYSTRAN has been in operation since 1969, when the U.S. Air Force first started using it to translate Russian scientific and technical texts. Since then, the Air Force has come to depend on it more each year; NASA now uses it for the translation

support activities (English to Russian and Russian to English) so vital to the success of the Apollo-Soyuz docking maneuvers. Despite the fact that computer translations may still sound stilted in places, NASA and the Air Force rely on this system's speed (300,000 words per hour) and its ability to produce consistent translations which accurately reflect the information content of original texts.

II.

This chapter describes SYSTRAN as it is today, and as it performs today. The expounding of theories of a hypothetical, ideal system which might exist someday is left to others—who may build one, someday.

Because SYSTRAN translates from more than one language (Russian, Chinese, German, English), the term *source language* is used to denote the language translated *from,* and *target language* to denote the language translated *to.* All the languages SYSTRAN translates have some basic things in common. SYSTRAN expresses the presence of these commonalities with the same codes, regardless of language. Thus one might say that the system uses a common denominator for all source languages. The simplest examples of common properties of languages are

1. Words to express action or a state of being (or equivalence: *"X* is *Y*").
2. A relationship between the action and that which performs the action ("John hits").
3. A relationship between the action and that which is affected by the action *("(X)* hits Mary").

Of course, all languages use vocabularies or dictionaries, even if they are not written. The first thing SYSTRAN does when presented with a text[1] is to look the words up in its dictionaries. Random access and sequential lookup procedures are combined to make computer use maximally efficient. For example, all idioms are selected on a longest match basis, using sequential lookup, whereas grammar codes are found via random-access procedures. Techniques using the attachment of addresses only and the attachment of all dictionary information are also combined. High-frequency words are given the

address of their dictionary information, whereas low-frequency words are supplied with the information itself.

Dictionary information is represented as codes[2], reflected by the appearance of specific bits in specific bytes. There are three types of codes:

1. Part of speech (signifying noun, particle, preposition, etc.) and subdivision within part of speech group (reflexive finite verb, adverb which may function as predicate, etc.);
2. Syntactic properties (allowable inflection pattern, object-type requirement, ability to govern infinitive, ability to govern preposition, ability to function impersonally, etc.);
3. Semantic features (human, combustible, sound device, malleable, measurable, mechanical process, etc.).

Although translation processing begins with dictionary lookup, the whole translation procedure is initiated at the moment the text is read into the computer. INITCALL, SYSTRAN'S master control program, calls for the execution of the dictionary lookup programs and calls all subsequent programs. There is no human intervention; INITCALL relinquishes control only when the translated text is ready to be printed.

Once the text words have been looked up in the dictionaries[3], those that comprise idioms will have been given their idiomatic meaning, those that are homographic will have been given a homograph type identification, and contiguous declinable words that have special meanings when in sequence will have been recognized and given appropriate meanings.

The first set of structural analysis routines (called PASS1) is called after PASS0 has established the function of all the homographs in the sentence. (Homographs are words like stand in English, which can either be a verb—stand there—or a noun —a stand of trees.)

Each analysis (and synthesis) program depends on the fact that each word in the sentence has, in the analysis area in core, 160 bytes for information about that word. Some of the information is supplied by the dictionaries; other information (a far greater percentage) is supplied by analysis and synthesis routines. To say that a word has 160 bytes reserved for it means that there are 160 spaces that can hold information. Each byte (space) can be thought of as being divided into 8 bits; any combination of these may be used to signify specific types of

information. Each program also depends on the fact that INITCALL has assigned each word in the sentence a sentence sequence number (a number identifying each word as the first word, second word, third word, etc. of the sentence).

PASS1 moves word by word through the sentence, turning on switches so that the location of nouns, verbs, and so on can be remembered, and establishing basic syntactic relationships[4]:

1. Noun + adnominal genitive
2. (Adverb +) adjective + noun
3. Adverb + verb
4. Verb + object
5. Preposition + object
6. Governor + infinitive
7. Governor + subordinate clause initiator (e.g., ČTO or ČTOBY in Russian).

The establishment of these syntactic units is reflected by byte relationships. That is, byte 18, for example, of the verb will contain the sentence sequence number of its object. Its object, in turn, will contain the verb's sequence number in byte 28. Thus, for SYSTRAN, an 18-28 byte relationship signifies a [governor + governed word] syntactic relationship. All syntactic relationships have equivalent SYSTRAN byte relationships. The syntactic signification of any byte relationship remains the same, no matter what language is being translated.

To establish syntactic units like those listed above, PASS1 depends heavily on the syntactic codes supplied by the dictionary for each word. For example, in determining whether a given Russian noun is the object of a particular preposition, the routine checks the byte information for the declension the preposition may govern. If the declined noun meets the preposition's requirements, and all other syntactic requirements are met in the context, government is established.

PASS2 extends the relationships established in PASS1 by recognizing enumerated noun modifiers, enumerated objects, enumerated adnominal genitives, appositives, and the like, and by establishing the function of enumerative commas and conjunctions, as well as the function of commas that set off parenthetic phrases.

At this point the full extent of prepositional phrases, noun

phrases, and verb phrases has been defined. In other words, nouns have been linked with all their modifiers (single word modifiers, enumerated single word modifiers, participial phrase modifiers), their appositives, their adnominal genitives, relative clauses for which they are head nouns, and the like, and verbs have been linked with their objects, modifiers, modals, auxiliaries, and the like.

Therefore, it is appropriate that the program which determines types and extent of clauses be called. This program establishes whether clauses are main clauses or subordinate clauses, marks each word in the clause with type indicator (first main clause, third main clause, relative clause, that-S complementizer clause, for-to complementizer clause, etc.), and sets up byte pointers to allow ensuing subject-predicate searches to "jump over" embedded clauses.

The subject-predicate search routines depend on the clause boundaries, types, and pointers established by the preceding program to limit the extent of their search and also to find all sections of interrupted clauses: the subject and predicate of any clause must be *within* that clause. Clause type indicators are important for determining types of allowable predicates. In Russian, for example, ČTOBY clauses need not contain finite verbs; infinitives suffice.

Subject-predicate searches must also depend on dictionary information. Number, gender, and case information is important for determining potential subject candidates and for matching proper subject candidates with predicates; codes attached to verbs may indicate, for example, that the verb is always used impersonally and never takes a subject or that the verb's subject usually follows, rather than precedes, it.

The above discussion briefly highlights the major functions of the analysis phase of translation processing. Further analysis routines determine the function of prepositional phrases and handle special structures idiosyncratic to the source language (such as "ratio" expressions in Chinese and the passive transformation in Russian).

When the analysis phase is completed, synthesis into the target language begins. These routines not only supply proper source-language inflections (or preposition insertion to denote case, such as to for the dative in English) and article insertion, but also perform conversions such as [VERB + $ADJ_{(OBJ)}$]=>

[VERB + THAT WHICH + $VERB_2$] where $verb_2$ is reconstructed from the adjectival form. These functions, again, depend on the presence of proper dictionary codes for allowable endings, restrictions on article insertion, cross-reference verb stems, etc.

After lexical synthesis has been effected, sentential constituents are rearranged to conform to target-language word-order requirements. In translation from Russian to English, the most frequently necessary rearrangement procedures are

1. Converting OBJECT + VERB + SUBJECT to SUBJECT + VERB + OBJECT order.
2. Converting MODIFIER + PREPOSITIONAL PHRASE + MODIFIED WORD to MODIFIER + MODIFIED WORD + PREPOSITIONAL PHRASE.

III.

It is simple enough to say "PASS1 does this, PASS2 does that, PASS3 does something else, ..." Most people, however, would probably want to ask *how* computer programs can do these things.

There is certainly nothing magic or mysterious about it. As we pointed out earlier, any rule that can be precisely formu lated can be programmed. Hence the most difficult part of the job is uncovering the rules that govern languages. We know these rules exist; if they did not, everyone would speak his own version of the language, and no one would be able to under stand anyone else.

Another indication that these rules exist is the fact that we can understand what a sentence is "talking about," even if we don't know the meanings of all the words. For example, in the sentence

Gorly gruk gobbers rogurtted the tobers, blikking mewsp in a pousap botorov.

we know that gobbers, which are gruk, even gorly gruk, did something: they rogurtted something(s) called tobers. At about the same time they did the rogurtting, something called a mewsp was blikked in something called a pousap

To a computer all sentences are like this. Neither the ma-

chine nor the syntactic analysis program knows the meaning of any word, even though a dictionary lookup program at-taches target-language meanings to source language words. Rather, the programs called into the computer's memory look for codes attached to the words. When looking at any particu-lar word to examine its syntactic environment the programs look for specific codes on the other words in the sentence. Such words could supply the following information:

1. "Gorly" is an adverb.
2. "Gruk" is an adjective/noun homograph.
3. "Gobbers" is a nominative noun.
4. "Rogurtted" is a verb (third person plural, past perfective) that requires an accusative object.
5. "Tobers" is a noun and may be either genitive if singular or nominative or accusative if plural.
6. "Blikking" is a present active participle formed from the verb blik, which requires an accusative object.
7. "Mewsp" is an adjective, either accusative or genitive.
8. "In" is a preposition requiring either an accusative or locative object.
9. "Pousap" is a locative noun.

SYSTRAN would decide in PASS1 that "gorly gruk" modified "gobbers"; that "tobers" is the object of "rogurtted" and is, therefore, accusative plural; that "pousap" is the object of "in"; that "mewsp" does not modify "botorov", but is the elliptical object of "blikking"; that "botorov" is the adnominal genitive of "pousap."

Had it been the case that "blikking" required a genitive object, SYSTRAN would have established that "mewsp" modified "botorov," that "botorov" was the object of "blik-king," and that the prepositional phrase "in a pousap" had to be postposed after "botorov."

In other words, SYSTRAN depends on grammar codes sup-plied by the dictionary and on the position of lexical items relative to each other. Word order is always an important consideration, but it must be remembered that word order rules vary from language to language. For example, if the nonsense sentence given above were Russian, structural analy-sis programs would be written to expect the preposition's ob-ject to follow the preposition and to expect an adnominal geni-tive to follow its head noun. On the other hand, if the language

were Chinese, the preposition might be the type that follows its own object. In addition, in Chinese, adnominal genitives precede their head nouns, the structure being marked by the particle *de.*

Grammar codes and word order are not always sufficient. When source language words are polysemous (i.e., have two or more different meanings), their different meanings will require different target language equivalents. For example, the Russian word TSEL' may be translated as target or as purpose (or objective, etc.). In order to determine which meaning the word is intended to convey, it is necessary to examine the word's semantic environment and the syntactic relationships between the word in question and other words that may affect or determine the choice of a target-language equivalent. To do this, SYSTRAN programs look at semantic codes attached to the polysemous word and to words that have the potential to influence its target-meaning selection. For example, a projectile may be fired at a target, but not at a purpose. A phase of a mission may have a purpose, but not a target (except, perhaps, figuratively).

At present, the system has multiple-meaning disambiguation routines only for those words that were chosen to be included in a semantic study effort. These routines are called after the structural analysis routines, in order that they may depend on previously established syntactic relationships. As time permits, more and more source language polysemous items will be included in semantic analysis routines.

On the other side of the coin, the use of semantic characteristics cart be a great aid in disambiguating syntactic structures. Here, again, though, the surface has barely been scratched. To a limited extent, SYSTRAN'S structural passes include semantic tests for allowable relationships. For example, a structurally plausible [head noun + adnominal genitive] may be disallowed because establishing such a relationship would create a semantic anomaly.

IV.

As we said earlier, a good computer translation system depends on proper linguistic analysis. Perhaps it also should be pointed out that it is essential that the results of the linguists'

work be accurately reflected in the programs. Generally, qualified programmers are not trained as linguists; by the same token, qualified linguists are not trained as programmers. This situation gives rise to the possibility of a serious communication gap: linguists investigate problems and devise rules that are meaningful to them; unless the programmer, however, knows exactly what the rules are supposed to do *(i.e.,* what type of language problem necessitated the rules and what the desired effect of the rules is), there is every chance that something will be lost in the "translation" of linguistic rules into programmed instructions.

For that very reason, SYSTRAN'S systems analyst devised a set of macro instructions that comprise a natural-language-oriented programming language. These macros are mnemonic, look like English, and sound like the procedures a linguist might use.

For example, if you want to look for a particular word, type of word, or type of grammatical feature in the sentence, you scan the sentence: SCANL means scan to the left (of some fixed point you choose in the sentence); SCANR means scan to the right. To limit the part of the sentence you want to look over, you use "MAX" (maximum) operands, specifying the beginning or end of the sentence, the clause, particular number of words, previously fixed point, specific word, and the like. To ask questions about a property of any word in the sentence, TEST instructions are used. CMPWD is used to ask if a word is specifically X (red, house, walked, etc.). Other instructions ask how the word ends or begins.

SYSTRAN'S linguist finds this programming language easy to learn because its instructions do not seem arcane and because they represent the types of operations he wants to effect. Moreover, when a linguist finds the need for operations for which no macro instruction exists, he simply presents the problem to the systems analyst, who creates a new macro.

In this way the potential problem of a communication gap is obviated, because the linguist does his own conversion of linguistic rules into programmed instructions. We have found an ancillary benefit in this approach in that the linguist who programs learns how precise his rules must be before they can be programmed and implemented with efficacy. Because of this,

he soon learns to limit his rules to environmental restrictions so far as is necessary. That is, when his rules are context sensitive, as opposed to context free, the sensitivity and the character of the context must be programmed into the rules via as many "tests" as necessary.

SYSTRAN helps the linguist at his work in another significant way: diagnostic programs can be used to extract from extensive corpora of "live" text any number of sentences containing the specific type of problem (defined by structure, by co-occurrence of specific words or classes of words, etc.) he is working on. A linguist usually feels more confident about his analysis when he can examine 500 "actual" sentences instead of 25 or 50 he made up himself. Thus he simply has SYSTRAN do his field work for him.

V.

At this point, you may be asking yourself either "Why isn't computer translation more widely used?" or "If it's been around since 1959, but still hasn't become popularly used, why talk about it now?"

The answer to the first question is simple: machine translation is economical only for large amounts of translation. Therefore, only those like the Air Force and NASA, who need to have thousands and thousands of words translated, find it cheaper to use machine translation. However, it might be worthwhile to note that if many people or groups with lesser translation requirements organized a translation center, then computer translation could also be economical for them.

The second question cannot be answered so readily. First, as pointed out earlier, machine translation of the late 1950s and early 1960s was limited to second-generation computers. Not only were these slower than the ones we use today, but they had a lot less memory. The latter factor meant that analysis programs had to be much smaller than now. That meant chopping programs into smaller parts, which could severely affect programs that operate most effectively when they are able to keep track of everything in the sentence (for example, PASS1 and the subject-predicate program, discussed earlier).

More than that, linguistics has come a long way since the 1950s as has the position of the linguist in machine translation.

The SERNA system and others of the same genre and/or era, were inherently weak because of these factors. The Georgetown system did have several small "passes" (where SYSTRAN has PASS1) to establish basic syntactic units; it depended on input structures fitting into programmed sentence patterns which were used almost as templates, whereas SYSTRAN depends on linguistic generalizations about how languages tend to behave. The systems and programming staffs were not linguists; the linguists were not programmers. Hence when the computer translation required modification, no one knew exactly where the changes should be made. The system became a black box whose inner workings could not be modified as new facts about language were uncovered because no one dared put his hand in.

This is not to say that this system sprang full-grown from an omniscient creator's forehead. Rather, the system grew out of the 1960s, benefiting from the work spent on its predecessors, and from the constant effort expended in overcoming its own problems. Easily amenable to change, it is always being changed, as new techniques of software management and better linguistic analyses are implemented to improve translation quality.

FOOTNOTES

1. Texts are inputted either on keypunched cards or magnetic tapes. No text is ever preedited or modified in any way other than by standard transliteration (Romanization).
2. Naturally, all these codes are supplied to dictionary entries by (hu man) linguists.
3. The plural is used because SYSTRAN employs specialized dictionar ies as well as a general dictionary.
4. The routines that establish these relationships are called by the part of speech of each word in the sentence.