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

JULES MERSEL

Programming aspects of machine translation

During the years that have elapsed since high-speed digital computers first became generally available, knowledge about the art of programming has been acquired by many groups. The first part of this discussion is intended for those readers who have not had an initial exposure to programming.

Programming can be considered as being similar to writing instructions for a girl who is working with a desk calculator. The analogy becomes exact if you assume that such a computress has the virtues of being very quick, of never making mistakes, and following instructions perfectly—and that she has the glaring defect of being incapable of thinking.

If it were required that our computress solve a problem, a notebook of instructions could be prepared for her. She would be told to go to line 1, page 1 of that notebook. There she might find an instruction which reads *Add the number on line* 6, *page* 73, *to the number found on line* 19, *page* 12, *and place the result on line* 13, *page* 62. *After you have done that, go to line* 3, *page* 9, *to find out what to do next.* There she might find a similar instruction. The differences in the instructions would involve simply a change in line and page number and a change in the arithmetical operation to be performed. Only occasionally would she be called upon to make a decision. The decision would be as trivial as *Look at the number on line* 12, *page* 3, *and compare it with the number on line* 24, *page* 22. *If the first number is bigger, go to line* 16, *page* 36, *to find out what to do next. If the first number is not bigger, go to line* 9, *page* 7, *to find out what to do next.* Eventually she would find an instruction which said *Hand the notebook back to me; your task is finished.*

Programming for a computer is no more and no less than the writing of such a notebook. The instructions available to the programmer, though more extensive than just the arithmetical commands, are few and quite simple. The additional instructions consist of such things as writing onto magnetic tape, reading from magnetic tape, and performing certain simple Boolean operations. Unfortunately there are no such instructions

as *be intelligent, do the right thing, jump to a conclusion,* or *find an adjective.*

As can be seen from the above analogy, digital computers do not think; they merely do the mechanical part of the white-collar work. In order for any problem to be solved on a computer, it must be well stated. Such instructions as *supply the appropriate preposition* (and this did appear on our original syntactic flowcharts) must be amplified so that the digital computer is told very specifically exactly what to look for, by what characteristic to recognize it, and then as a result of its search to decide exactly which preposition must be supplied and where it must be inserted.

Writing such a notebook is a time-consuming and tedious task. However, it would take less time to perform the desired operations manually than to program them. Digital computers are advantageous only when one is going to perform the same task over and over again. Then the speed of the computer makes its application economical. As an example one might give the problem of finding the sum of a million numbers. One wouldn't say, as one might in hand calculations, *add the first number to the second number, add the third number to the sum, add the fourth number to the sum,* until one finally said *add the one-millionth number to the sum.* Instead, one would do something logically more complex and fantastically shorter. The instructions would be:

Set the sum to equal 0.
Set i to equal 1.
Add the ith number to the sum.
Does i = 1 *million? If so, you are finished. If not, increase i by one and proceed through the third and fourth instructions.*

We now can begin to get a glimmer of how a computer can be useful. It wouldn't take the computer long to do such a computation, provided the programmer can find a short way of telling the machine explicitly how to do it.

In recent years compilers have been developed that simplify the task of writing programs. One such program, COMIT, was specifically written for language-data processing. However, even with compilers, the detailed spelling out of instructions is not avoided—it is merely raised to a less dreary level.

INPUT AND OUTPUT

Our computress analogy can be expanded to embrace the notion of input, output, and erasable storage. Let us assume that instead of giving our fictional computress just one notebook, we gave her four such books.

One of the books would be the same as the kind described above-it would contain the program and no numbers would be written into it by the computress.

Another notebook, which we can call the input, would contain all the numbers which change from problem to problem and which are used in this calculation by the computress.

A third notebook, the output, would originally be blank and would eventually contain just those numbers that the problem-giver eventually wanted to see. The fourth notebook, the erasable storage, would contain all the intermediate results and modified instructions that were needed to arrive at the output. In our computress—desk calculator analogy, this fourth notebook would be thrown away when the problem was solved; in computing machines, the erasable storage is part of the equipment, and is merely wiped clean so that new information can be written over it. The point to be remembered in machine translation, or for that matter in any other type of research, is that if the intermediate results will be needed for future use, they must be placed into the output. If they are left in erasable storage they will be erased, and the only way of retrieving them would be to go through the whole problem again and record from erasable storage before erasure takes place.

TYPES OF COMPUTER STORAGE

In computing machines, storage, or memory, can consist of three types: The first type is *random-access* storage. Random-access storage is typified by the speed with which one can write into or read from any storage location. This speed is based on the equal availability of all storage locations. The time required for such reads and writes varies from a fraction of a microsecond (1 μsec = 0.000001 second) for evaporative-film memory to 20 μsec for the slower magnetic-core memories.

A second type of memory is *serial* memory. Serial memories require positioning for either writing into or reading from them. A common type of serial memory is the magnetic drum. Reading or writing can be done only when the drum has turned to a certain position. Waiting for this to occur can take as much as 32 milliseconds (1 msec = 0.001 second). Though this is a far longer wait than would be required with even the slowest core storages, the drum has the compensating advantage of providing much more storage per dollar expended.

The third type of storage is exemplified by *magnetic tape*. Magnetic tapes can hold extremely large amounts of information. Unfortunately if the information is far from the reading head of the tape unit, minutes may elapse before the information becomes available. To a large extent, these long waits can be avoided by careful programming. Tapes have the advantage of being inexpensive: a reel of tape costs $50, machine time may cost $400 per hour. They also can be removed from the computer for later use: output can be written onto them, and later input can be read from them. This input-output use is one of the main advantages of

magnetic tapes; the ability to save intermediate results is important economically.[1]

DICTIONARY SEARCH

The programming of any particular problem is affected almost as much by the characteristics of the available computer as by the nature of the problem itself.

In the following discussion we shall consider the programming techniques used by three different machine-translation groups to solve the problem of searching a Russian-English machine dictionary.

If we ignore the question of what a dictionary should contain, then the best-defined portion of the machine-translation process is the dictionary lookup. In concept, nothing could be simpler. Given a word from text, look it up in the dictionary. If we had an ideal machine with an infinite and instantaneous-access memory, we would merely take the Russian word, use it to find our item in the dictionary, and be through with the lookup. Unfortunately such a machine does not exist. The problem is complicated by the limited amount of fast storage available on computers today.

The random-access memories on present general-purpose computers do not have room for a complete bilingual dictionary with all the necessary information.

Though a dictionary can be ordered (the most common ordering being alphabetical), natural text comes in a completely unpredictable and random order. The problem therefore is that if only part of the dictionary is placed in storage, there would be no assurance that when the next text word is selected, the required portion of the dictionary would be easily available. The solutions to this problem reflect the interests and goals of various machine-translation groups.

One solution to the problem is that used by the group at Thompson Ramo Wooldridge Inc. This group uses a computer with good computational ability, a small but fast internal storage, and excellent magnetic tapes. The computer can transfer information to and from magnetic tapes while performing other computations.

The dictionary is stored in alphabetic order on magnetic tape. Four thousand words of text are transferred to internal storage. These text words are sorted into alphabetic order. The beginning of the dictionary is then brought into the internal memory. The alphabetically ordered text words are now compared with the dictionary items that are in fast storage.

This comparison does not simply follow the alphabetic order. Instead,

[1] Since the time this paper was given at the UCLA lecture series, disk-type memories have become an important type of storage medium associated with high-speed computers.

a binary search is performed. The text word is compared to the last word of the dictionary portion in fast storage. If it is alphabetically smaller, it is compared to the middle word of the dictionary; depending on this last comparison, the text word is then compared to the word one-quarter of the way through or three-quarters through the dictionary portion. Thus, the amount of dictionary to be compared is halved at each test. Consequently, looking at a table of 1,024 words requires at most ten comparisons rather than the average of 512 that would come from a straight search in alphabetic order.

As soon as it has been determined that a word of text is alphabetically beyond the part of the dictionary that is in memory, that part of the dictionary is replaced by dictionary items further down the alphabetic list. This is done simultaneously with dictionary lookup.

In order to allow that portion of the dictionary which is in memory to include as much of the whole dictionary as is possible, very few forms are carried for any one word. Thus there is a high likelihood of a text word not finding a representation in the dictionary.

When a text word is not in the dictionary, the word is analyzed morphologically; that is, an attempt is made to separate the stem of the word from its ending. This process of stem-ending analysis requires a good computer and an extensive program. It is aided by storing a list of possible endings in memory. Dictionary words surrounding the place where the text word should have been found are then split into stem and ending. If a dictionary word and a text word have the same stem, the endings are compared to see whether the ending of the text word will predict a grammar code compatible with the grammar code of the dictionary word. If this is the case, then the text word is given a revised grammar code derived from the grammar code of the dictionary word as adjusted on the basis of the ending of the text word. The English equivalents associated with the dictionary word are reinflected so that they represent the tense, number, or person corresponding to the form of the text word. The text words are thus replaced by the corresponding dictionary entries, revised wherever necessary by stem-ending analysis.

The lookup process is continued until all 4,000 text words have been examined. Then the beginning portion of the dictionary is brought back into internal storage and the next 4,000 words of text are looked up by the same method.

When the complete text has been examined, the dictionary items corresponding to the text words are restored to text order. The text is then examined in order to identify contiguous idioms.

The batching method just described utilizes the speed of the computer to enable it to supply missing forms in -less time than would be required to spin a tape which contained all the forms. Another method, in use at the University of California (Berkeley), was programmed for a

machine with good computational ability, but no ability to spin tapes during computation. Here an attempt is made to get a very large dictionary into memory. Since the memory itself is not very large, this is ingeniously achieved by dividing the dictionary items so that the Russian word is completely separated from the other parts of the dictionary entry. All of the Russian portion of the dictionary is brought into memory at one time. Space is conserved by eliminating the first two letters of each word and indicating in a table where each two-character list starts. Space is also saved by making the dictionary purely a stem dictionary. The Berkeley group estimates that it has the capability for storing 20,000 stems.[2]

As each text word comes in, its first two characters are examined to find the appropriate two-character list. Then the remainder of the word is looked at in successive locations of that portion of the dictionary. The dictionary items are not stored in alphabetic order but rather in order of length. When a dictionary item is found whose characters exactly match the beginning characters of the text word, a similar search is instituted to find out whether the remaining portion of this text word also has a dictionary stem equivalent or a dictionary ending equivalent. Thus, in this dictionary, the word *microorganisms* would be found in three searches: one for *micro,* one for *organism,* and one for *s.* After the word has been identified, the locations of the dictionary items pertaining to this word are stored in successive order. Since this redundant information can be stored in comparatively little space, the Berkeley system allows passing approximately 30,000 words of text through the dictionary before the storage space is filled up. When the examination of the Russian portion of the dictionary is completed, the next segment of the dictionary is transferred to internal storage. Here, no further search is involved. For each text word, the correct segment can be picked up right away and placed onto tape. This is continued until all segments of the dictionary have been brought into memory. This technique allows the Berkeley group to claim the fastest lookup of any of the dictionary search methods. Its drawbacks are that there is no guarantee that the Russian portion of the dictionary will not eventually grow too large for storage, and the dependence on the assumption that the correct translation of a word can be put together from the English equivalents of its separate parts.

Another type of dictionary lookup has been constructed at the RAND Corporation. Here again, there is an attempt to get the whole dictionary into the memory. The whole dictionary, however, does not consist of the

[2] Since the time this paper was given at the UCLA lecture series, Sydney Lamb, head of the Berkeley group, has made major modifications to his initial dictionary concept. These now allow his research group to state that its technique will allow for the handling of a far larger lexicon. Similar improvements have been made by other groups.

Russian lexicon but rather of the lexicon that is needed for the first portion of text. The size of this first portion is determined by the nature of the text itself. Two passes at the text are made. The purpose of the first pass is to determine how many text words can be handled by a dictionary which fills the available space in high-speed storage. Recognition is made of the fact that 4,000 words of dictionary may easily be sufficient to handle 100,000 words of text. Thus, after the first pass is made, the RAND program knows exactly what words from the dictionary need to be in high-speed storage. These dictionary items are brought in and the text is passed again. This double pass at the text eliminates the problem of re-sorting and thus considerable efficiency is effected.

Another solution to the problem has been attempted at IBM. Here, as befits a computer manufacturer, the solution has been to build an internal storage large enough to hold an extremely large memory. This storage is a photoscopic disk. It is a serial storage that is read by optical techniques rather than magnetic sensing. The storage cannot be written on during computation. Since the storage capacity of the disk is quite large and the period of rotation short, text is not sorted.

The capacity is large enough to allow storage of phrases. The lookup proceeds by phrases; single words are treated as "degenerate" one-word phrases. This storage of idioms has allowed for a fairly extensive but slow dictionary search.

In summary, we can see that though the problem of looking up items in the dictionary is conceptually easy, the configuration of hardware either requires the use of less than straightforward methods, or has inspired at least one group to build special hardware.

SYNTAX

Programming the syntax portion of machine translation differs from the dictionary portion in that words on an individual basis are not the primary concern, rather certain grammatical characteristics which a word may share with thousands of other words. The grammatical characteristics are recorded in the grammar code for the word. If the goal were the maximal conciseness of the grammar code, it could be compressed into just a few alphabetic characters. Indeed, the grammar code that the Ramo Wooldridge group first started working with (it was borrowed from the RAND Corporation) consisted of three alphanumeric characters. The decoding of these alphanumeric characters for use in the algorithm, though possible, is not convenient. TRW and Wayne State University, therefore, chose a binary representation which C. Briggs of Wayne[3] referred to as the vectorial representation of the word.

[3] Personal communication.

Figure 1 shows the grammar code for a modifier. It is called a *modifier* grammar code rather than an adjective grammar code because in our representation of the parts of speech (word classes) of Russian the modifier class includes not only adjectives, but also participles and possessive pronouns.

The grammar code consists of 108 binary digit positions, shown in the figure in 3 columns of 36 each. Each position contains either a one or a zero. It allows no other representation.

The first column of the grammar code has the same format for all parts of speech. Positions 1 through 9 indicate the word class of the word. Only one of these positions will contain a one; all the others will contain zeros. In our example, bit position 3 is occupied by a one. It is reserved for indicating modifiers.

In column 2, bit positions 11 through 36 are labeled the *agreement code*. They represent the possible genders, numbers, and cases that a given word may assume: *M* stands for masculine, *F* for feminine, *N* for neuter, *P* for plural, *S* for nominative, *G* for genitive, *D* for dative, A_i for accusative inanimate, A_a for accusative animate, *I* for instrumental, and *L* for prepositional. Unlike the word-class code, the agreement code and the government code of the third column do not have the constraint that only one position is allowed to contain a one. As many ones may be used as are needed to describe all possible features of gender, number, and case of a given word, or of the gender, number, and case of the words that this word might govern. Note that this requires many more ones in the government code than would be needed if we were merely to indicate by a single symbol the possible cases that a word could govern. Note also that the binary agreement code requires more storage than a more condensed representation such as the earlier one consisting of alphanumeric characters. The reason for this lack of compactness is the great convenience afforded by the binary codes in agreement and government checks.

This will become apparent after we take a brief digression into the area of Boolean logic. As indicated by Maron earlier in "A Logician's View of Language-data Processing," $A \wedge A = A$; $A \wedge B$ indicates the common portions of *A* and *B.* In the computer when doing a Boolean operation, $1 \wedge 1 = 1$, $1 \wedge 0 = 0$, $0 \wedge 1 = 0$, $0 \wedge 0 = 0$. In the grammar code, the presence of a one in a particular bit position indicates that the gender, number, case for which this bit position is reserved, applies to the word in question.

In a government check, the government code of the governing word is compared to the agreement code of the potential governed word. This comparison is done by a *logical-and* operation. The result of this operation is that those bit positions which in both grammar codes are filled by

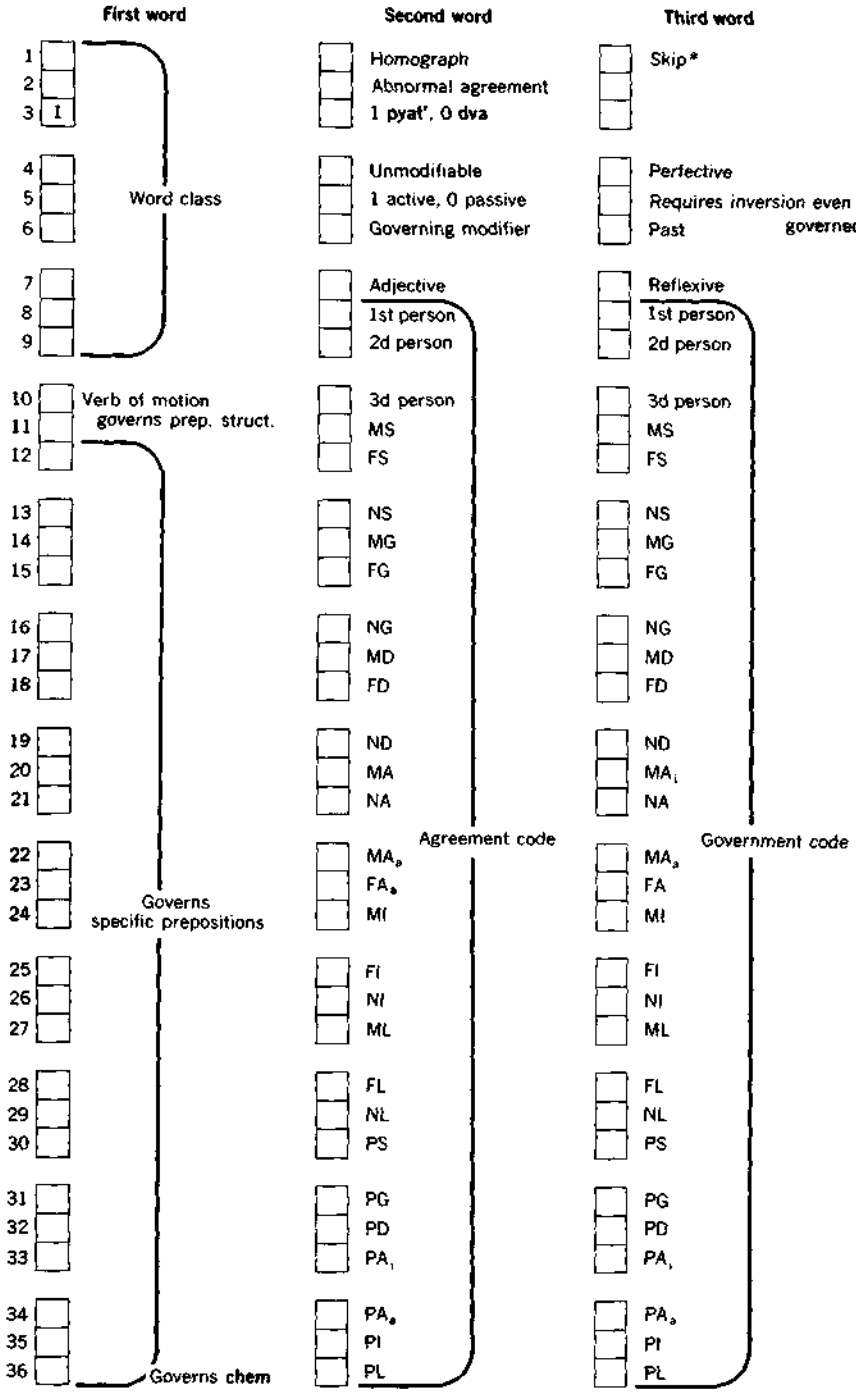

First word
Second word
Third word
1
2
3 I
Word class
4
5
6
7
8
9
10
11
12
Verb of motion governs prep. struct.
13
14
15
16
17
18
19
20
21
22
23
24
Governs specific prepositions
25
26
27
28
29
30
31
32
33
34
35
36
Governs chem
Homograph
Abnormal agreement
1 pyat', 0 dva
Unmodifiable
1 active, 0 passive
Governing modifier
Adjective
1st person
2d person
3d person
MS
FS
NS
MG
FG
NG
MD
FD
ND
MA
NA
MA$_a$
FA$_a$
MI
FI
NI
ML
FL
NL
PS
PG
PD
PA$_1$
PA$_a$
PI
PL
Agreement code
Skip*
Perfective
Requires inversion even without governed struct.
Past
Reflexive
1st person
2d person
3d person
MS
FS
NS
MG
FG
NG
MD
FD
ND
MA$_1$
NA
MA$_a$
FA
MI
FI
NI
ML
FL
NL
PS
PG
PD
PA$_1$
PA$_a$
PI
PL
Government code

FIGURE 1

ones remain filled by ones in the sum. This indicates that there is a relation of government in this case. Conversely, if the result is a string of zeros without any ones, then there are no ones in corresponding bit positions, and hence the first word cannot govern the second word. In an agreement check, the agreement codes of two potentially agreeing words (such as a noun and preceding adjectives) are compared by a logical-and. Those bit positions which contain ones in both agreement codes result in ones in the sums. Bit positions that are not occupied by ones in both codes, but only in one of them, result in zeros in the sum. These additional ones indicate ambiguities in gender, number, and case; their zeroing in the logical sum indicates a resolution of the ambiguity. The logical-and operation requires only one computer instruction.

Each column of the grammar code shown in Figure 1 represents a separate *computer word*—that is, a fixed grouping of memory locations. Each computer word of the grammar code can be tested separately. Using the logical-and operation on the appropriate computer word, a Russian word can be tested to determine whether it is a modifier. A second test can be used to determine whether it is a *governing modifier—that* is, a modifier which can govern a dependent structure (e.g., the word *important,* as in *important-by-comparison results).*

The question may well be raised whether one and zero (standing for *yes* and *no)* are sufficient for the grammar code of a Russian word. In our own case, a ternary system may be needed, where " + 1" indicates *yes,* "— 1" indicates *no,* and "0" indicates *I don't know.* Ida Rhodes of the National Bureau of Standards uses a quaternary representation for government where a "3" indicates that something is mandatory, a "2" that something is highly probable, a "1" that it is merely possible, and a "0" that it is impossible. Mrs. Rhodes goes even further and makes the code octal, which adds one more bit location. This extra bit location is used to indicate whether the condition must be fulfilled immediately or at any time within the clause.

As pointed out in Garvin's "Syntax in Machine Translation," the syntax-analysis method used in the Ramo Wooldridge machine-translation program is *multiple-pass.* The first pass consists of examining the whole sentence for numerals and symbols. Whenever a symbol or a numeral is detected, its neighborhood is examined in order to find the conditions for supplying a grammar code. The second pass again consists of an examination of the whole sentence, this time in order to isolate and resolve word-class homographs. Figure 2 shows one portion of the homograph-resolution flowchart. As the flowchart is entered, it has already been determined that the homograph has the potential part-of-speech functions of a predicate, an adverb, or a preposition. The start is at the circle labeled HR1. Each box on the flowchart represents a

question and can be answered by either *yes* or *no.* The circles, with the exception of those marked PP2, represent jumps to other parts of the same flowchart. PP2 represents a jump back to another routine which searches the sentence for further homographs.

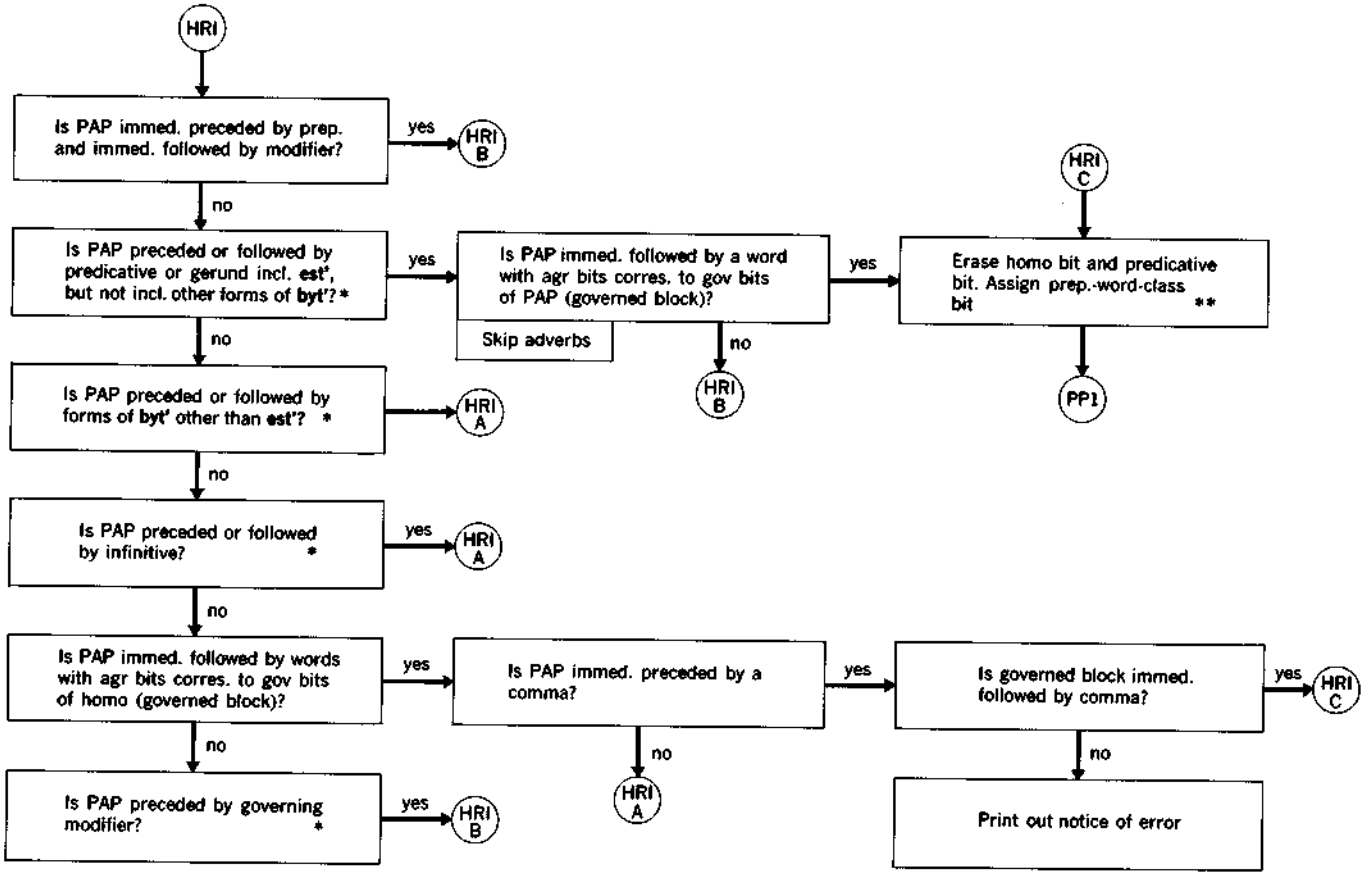
The next pass is devoted to the packaging of nominal blocks, which may consist of just a noun or a noun modified by one or more adjectives. The process of nominal blocking allows for the reduction of case, gender, and number ambiguities through a matching of the agreement codes, as discussed above.

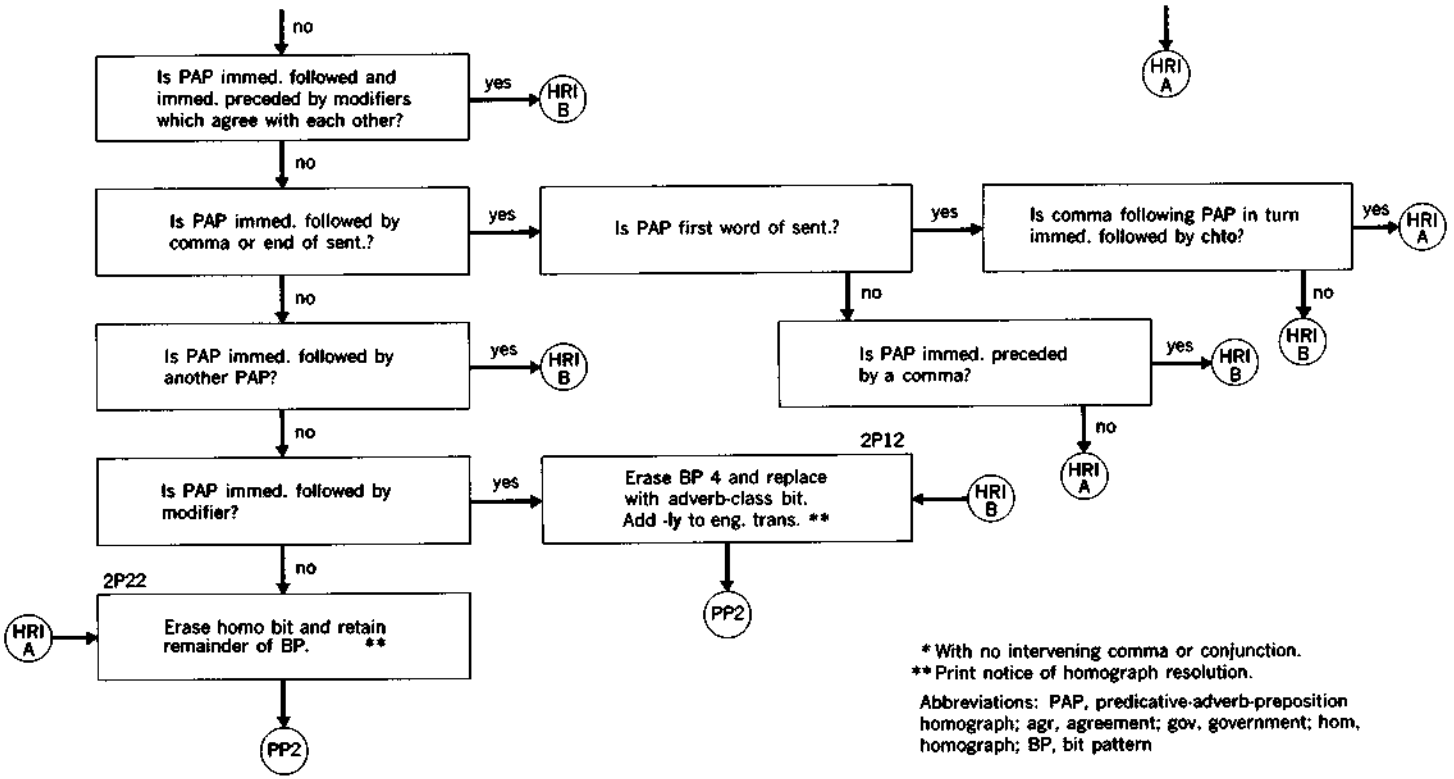
After the nominal blocking is completed, the next pass is devoted to the creation of prepositional packages, consisting of prepositions together with the words they govern. Here the government code of the preposition allows an opportunity for reduction of ambiguities, comparable to that afforded by agreement checks. After this pass the adverbs of the sentence are searched for and inspected to see whether they can be included in the previously prepared nominal packages. Following the adverb pass, the routine goes searching for inserted structures—these are structures which are independent of the remainder of the sentence, and once identified, they can be skipped over and ignored in the major syntax, which is discussed further below. The search for inserted structures requires a fairly close examination of the commas and other punctuation of the sentence.

After the inserted-structure pass, a search is made of the sentence for governing modifiers, which include mainly participles. These are examined to see whether they are acting merely as adjectives or whether they govern other word groups in the sentence. If rearrangement of the English is called for, indication of this rearrangement is made.

After the governing-modifier pass, the routine once again goes hunting for commands in order to determine where major clause boundaries can be definitely established. The next and last search preceding the major syntax is the search for relative clauses, which are identified and labeled *relative packages.* Up to this point we have dealt with packages which can be skipped over in the search for the predicate, subject, and objects of the sentence, and we have reduced certain ambiguities by inspecting individual words and including them in blocks where possible.

The major syntax consists of a search of the main clause (skipping over inserted structures, governing modifier packages, and relative packages) in order to find first the predicate of the sentence. Once the predicate has been found, this gives us certain information about the nature of the subject (certainly a plural verb will not have a singular subject) and about the possible object or objects of this predicate. Then a search is made to the left of the predicate for the subject. If one is not found and if the predicate is not of the type allowing an implied subject, our earlier


HRI
Is PAP immed. preceded by prep. and immed. followed by modifier?
yes
HRI B
no
Is PAP preceded or followed by predicative or gerund incl. est', but not incl. other forms of byt'? *
yes
Is PAP immed. followed by a word with agr bits corres. to gov bits of PAP (governed block)?
Skip adverbs
no
HRI B
yes
Erase homo bit and predicative bit. Assign prep.-word-class bit **
HRI C
PP1
no
Is PAP preceded or followed by forms of byt' other than est'? *
HRI A
no
Is PAP preceded or followed by infinitive? *
yes
HRI A
no
Is PAP immed. followed by words with agr bits corres. to gov bits of homo (governed block)?
yes
Is PAP immed. preceded by a comma?
yes
Is governed block immed. followed by comma?
yes
HRI C
no
HRI A
no
Print out notice of error
Is PAP preceded by governing modifier? *
yes
HRI B



* With no intervening comma or conjunction.
** Print notice of homograph resolution.

Abbreviations: PAP, predicative-adverb-preposition homograph; agr, agreement; gov, government; hom, homograph; BP, bit pattern

FIGURE 2

program would print out *notice of syntactic difficulty,* indicated by the alphanumeric code number 41 in the vertical output (see below). We no longer do so; we now determine which part of the clause portion that follows the predicate should precede it. This can be done successfully in a surprisingly large number of cases. Code 41 now implies a certain type of sentence rearrangement.

If a subject has been found to the left of the predicate, a search is made for the possible objects which the government code of the predicate indicates. This search will often detect whether or not an English preposition should be inserted in the final translation. If any difficulties have been encountered in the major syntax, the computer exits to a cleanup pass where as much of the clause as can still be identified is analyzed.

The syntactic analysis of the clause is completed, but the syntactic information that has been gleaned must be thoroughly examined. The results of the syntactic analysis permit the resolution of certain translation problems by means of a word-combination routine. This routine examines verbs and their complements, after their connection has been established by the syntax routines, in order to determine their correct translations on the basis of their joint occurrence in a sentence. The word-combination routine ascertains the meaning of certain verbs on the basis of their complements, and even in those cases where the meaning of the verb cannot be determined, the translations of certain of the prepositions (implied or otherwise) are selected. After the word-combination routine, the rearrangement of the English in the sentence is performed. After the rearrangement, the correct English equivalent for certain Russian words is selected on the basis of the Russian case number and gender which by now may have been determined; the definite article is inserted in front of certain packages, and in front of those genitive, instrumental, and dative packages for which a preceding preposition has not yet been inserted, the appropriate insertions are now made; finally, ad hoc multiple-equivalent rules are applied.

In the multiple-pass method just described the sentence is examined many times, each time to hunt for specific items. Another method, less resembling that which would occur to a linguist, is the method of *predictive analysis* developed by Mrs. Ida Rhodes of the Bureau of Standards. In her method, the sentence might only be examined twice. The first examination, which she calls the *profile pass,* has as its purpose to label which clause each word in a sentence is in. She allows for, and treats, nested clauses. The search for clause indicators is primarily a search for commas, conjunctions, and certain specific words. After she has created this clause identification, which she calls the *profile* of the sentence, she then is ready for the second and possibly last pass. Here, her technique consists of examining each word in turn and asking whether it satisfies

any predictions, and then asking what further predictions the presence of the word allows her to make. "Predictions" are assumptions of varying strength regarding the presence of certain grammatical conditions. If the assumption is borne out, the prediction is considered satisfied. She handles her words in the order of their appearance in the text, without backtracking to earlier words, but backtracking to earlier predictions. She enters each clause with two mandatory predictions that are contained in the program: A clause must have a subject and a clause must have a predicate. Hence, she can ask with the very first word of the clause, *Does it satisfy a prediction?* If a word satisfies a prediction, she crosses this prediction from her list. She examines her predictions in reverse order, that is, the last prediction made is the first one examined. This allows her not to go too far back within a clause in order to find the desired prediction. Certain nonmandatory predictions are wiped out if they are not satisfied by the word immediately following the word that brought them about. Unused predictions that have not been wiped out are stored in a "hindsight pool." Limits are placed on prediction: In a two-clause sentence, a nominative noun in the second clause cannot satisfy a prediction for a subject of the first clause. When a sentence has been completed, Mrs. Rhodes' method asks whether any mandatory predictions remain unsatisfied. If there are unsatisfied mandatory predictions, the hindsight pool is inspected for alternative possibilities to the resolution of the sentence. Using this hindsight pool, another pass is made at the sentence to see whether the mandatory predictions can now all be satisfied. Further passes at the sentence could, of course, be made until all of the possible translations have been obtained. Mrs. Rhodes prefers, however, to stop after the first acceptable analysis has been completed. In this respect, her technique is at least as good as the other translation techniques which allow for only one possible evaluation, and she has certainly created the tools to detect syntactic structure.

At first glance, her technique seems more palatable from the point of view of programming than a multiple-pass method. Her pass at the sentence is straightforward; it considers and treats each word as it appears. Her search for predictions is always backwards and terminates with the most recently made useful prediction. In some senses, however, it is not too much more straightforward than the multiple-pass method. The multiple-pass method consists of finding the word, noting its predictions, and searching the sentence for other words that might have been predicted by it. Mrs. Rhodes' technique consists of finding a word and Marching backwards through the sentence for other words that might have predicted it. Her technique approaches its maximum programming desirability when there is an extreme shortage of storage, for she handles each word in a systematic order and her search backward has a low proba-

bility of going to the very beginning of the sentence. Even then, she erases predictions after they have been utilized. Thus, storage requirements are reduced.

THE OUTPUT

Several machine-translation groups create two types of output: one simply to display the translation, and one for research purposes. The Ramo Wooldridge group uses a so-called "horizontal output" for the first purpose, and a "vertical listing" for the second purpose. The horizontal output consists essentially of lines of English words constituting the translation; whenever a Russian word has more than one English translation and the program has not made a selection, the English translations appear underneath each other, creating a multiple line of English at that point.

For research purposes, much more is needed than just the English words of the translation. The output shown in Figure 3 is a vertical listing displaying the information that was extracted by the program in the process of translation.

The lines consisting of periods and ones or periods and *x*'s between each line of text show the grammar code referred to in Figure 1. The difference between the ones and the *x*'s is that those grammar codes that contain a one stem from the dictionary lookup, while those containing *x*'s were either created or modified by the program.

The first column gives the text location of the individual words. Thus all this text comes from page F, lines 45, 46, 47, 48, and 49, and the numbers following the line number refer to the position of the word on the line. The next columns containing the numbers 40 and 41 on line 6 represent syntactic translation decisions. The group of six numbers following that column displays the syntactic analysis of the sentence; each digit indicates a particular syntactic condition that has been identified by the program. The next column represents the desired English word order: a number is assigned to each English word, indicating the position which it should occupy in the translation of the sentence (in this case, it is identical with the Russian word order). The column after that is a transliteration of the original Cyrillic. Finally, there is the English translation. If you draw a vertical line just to the left of the word GENERAL, you find the dictionary English to the right of that line. To the left of that line are words like *the, do, of the;* these are words supplied by the program. The asterisk on the lines with the words AMERICAN, BELIEVE, and PEOPLE, indicates that the exact Russian representations of these words were not in the TRW dictionary and that the grammar code and possibly the inflection of these words was supplied by the stem-ending analysis described above. The translation says:

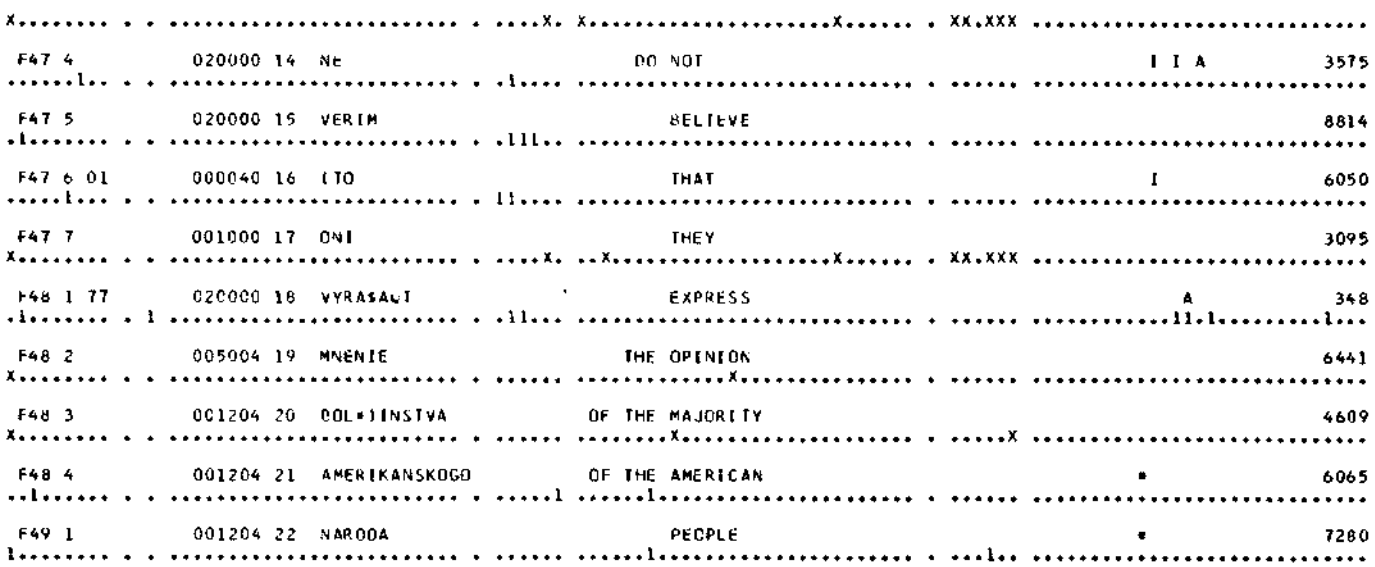

F47 4 020000 14 NE DO NOT I I A 3575
F47 5 020000 15 VERIM BELIEVE 8814
F47 6 01 000040 16 (TO THAT I 6050
F47 7 001000 17 ONI THEY 3095
F48 1 77 020000 18 VYRAŠAJ EXPRESS A 348
F48 2 005004 19 MNENIE THE OPINION 6441
F48 3 001204 20 BOL*JINSTVA OF THE MAJORITY 4609
F48 4 001204 21 AMERIKANSKOGO OF THE AMERICAN * 6065
F49 1 001204 22 NARODA PEOPLE * 7280

FIGURE 3

GENERAL PAUZR SENATOR GOLDVOTER TAKE (OR OCCUPY) THE IMPORTANT POSITION IN THE AMERICAN OBWVESTVE BUT WE DO NOT BELIEVE THAT THEY EXPRESS THE OPINION OF THE MAJORITY OF THE AMERICAN PEOPLE.

This, though far from being a perfect translation, is not ridiculous. A correct translation would read:

GENERAL POWERS AND SENATOR GOLDWATER OCCUPY AN IMPORTANT POSITION IN AMERICAN SOCIETY BUT WE DO NOT BELIEVE THAT THEY EXPRESS THE OPINION OF THE MAJORITY OF THE AMERICAN PEOPLE.

Studying the vertical listing, we found that the words GENERAL, PAUZR, SENATOR, GOLDVOTER, and OBWVESTVE had been missing from the dictionary. This was remedied by the program as follows: The analysis of the endings and of the neighborhood of the missing words by our missing-word routine indicated that GENERAL is a noun in the masculine nominative or the masculine accusative; that PAUZR, SENATOR, and GOLDVOTER are nouns with the unknown case, gender, and number, and OBWVESTVE is a noun in the neuter prepositional As shown in the vertical listing, appropriate grammar codes were created by the program and the syntactic analysis could proceed

We also found that the translation of the Russian word *I* as the conjunction *and* was missed, due to a glossary maintenance error that was committed just before this run. The vertical listing allowed us to detect this error.

Now to explain the vertical listing further. In the column representing syntactic analysis, the number 001000 indicates that each of the first five words (which all show this digit) are in a nominal block. Other numbers in the same column indicate that the sixth word is a predicate, the seventh and eighth words are nominal blocks that form an object, and the ninth, tenth, and eleventh words constitute a prepositional phrase, within which the tenth and eleventh words form a nominal block, that the thirteenth word is a nominal block, that the fourteenth and fifteenth words form a predicate, the sixteenth word forms a search boundary, the seventeenth word a nominal block, the eighteenth word a predicate, that the nineteenth word forms a nominal block and is part of two types of complement. This complement (used in the word-combination routine described above) extends from the nineteenth word to the twenty-second. The twentieth, twenty-first, and twenty-second words in addition constitute a nominal block, and the twentieth, twenty-first, and twenty-second are also in the genitive. The syntactic analysis has enabled the program not only to insert articles (some of which, as can be seen, are incorrect) but also the auxiliary *do* and the preposition *of* in two places. Without the syntactic analysis, the sentence would have read:

GENERAL PAUZR, SENATOR GOLDVOTER TAKE (OR OCCUPY) IMPORTANT POSITION IN AMERICAN OBWVESTIVE BUT WE NOT BELIEVE THAT THEY EXPRESS OPINION MAJORITY AMERICAN PEOPLE.

The code number 41 on line 6 indicates that, due to missing agreement codes, the routine was not sure it had identified the subject of the sentence. There was a possibility, as far as the program could see, of the first five words being the subject, and therefore no rearrangement took place.

The information presented in the vertical output is kept with other information on a magnetic tape that we refer to as the information tape. Each record on the information tape represents a sentence and each file an article. On each record, the first five and one-half lines contain the English information, the remainder of the sixth line contains information about stem length, the eventual English word order, whether the word was created by stem-ending splitting, whether the word participated in an idiom, and whether the grammar code was modified. Line 7 contains information about multiple-equivalent resolution. Line 8 contains information about the dictionary access number of the word, the form of the ending, and whether the word is potentially part of an idiom. Lines 9, 10, 11 contain the grammar code, lines 12, 13, 14, the Cyrillic. Line 15 contains information about the location of the word in the text and about surrounding punctuation, if any. Line 16 contains a record of Insertions to the right that the routine has decided upon. Line 17 and part of 18 are used for recording insertions to the left; the remainder of 18 and all of 19 are used for a record of the syntactic decisions. Consequently, it is possible to create routines that will go over previously translated text in order to search for relationships that were not thought of at the time of translation.

As to the size of the program: the dictionary lookup required 6,000 Computer instructions, and the syntactic analysis, 10,000 instructions. The dictionary lookup works at 180,000 words per hour and the syntactic analysis at 60,000 words per hour. That is, translation is done at the rate of 45,000 words per hour, or 750 words per minute, or 12.5 words per second.