THE DIGITAL DATA-PROCESSING PROBLEM

OF MACHINE TRANSLATION

OF RUSSIAN TO ENGLISH

BY

ROBERT E. WALL, JR.

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy,

University of Washington.

Introduction

This report is concerned with the application of digital data-processing equipment to the problem of the machine translation of language. In particular, the work described in this report concerns the translation of written scientific Russian into written English.

The research in Machine Translation at the University of Washington has been a team effort; hence the work of the author has been influenced considerably by the work of other members of the project. A brief history of the project is presented below in order that the motivation for the various investigations which are described in this report may be clearly understood.

Research in Machine Translation was started at the University of Washington in 1949 by Dr. Erwin Reifler, Professor of Chinese. The project has been sponsored since the spring of 1956 by the Rome Air Development Center. A complete history of the project may be found elsewhere;[1] here the work which has been accomplished will be outlined only briefly.

The Rome Air Development Center commissioned the project to prepare a translation lexicon of about 30 million bits, which would be suitable for word-for-word translations. This lexicon was to be entered into a photoscopic store then being developed by the International Telemeter Corporation.

The project first selected a corpus of approximately 30,000 words of scientific Russian text. This material was extracted from 111 different books and articles from about 40 representative fields of science. Each article was translated, and at the same time each Russian word was entered on a separate card along with the English equivalent for the word which was used in that particular translation. After this tabulation was completed, the cards were ordered alphabetically; and a single card was substituted for the several cards resulting from the fact that many Russian words had occurred more than once. On this single card was listed that particular word together with all English equivalents which had been used for that word. About 13,500 different Russian words occurred in the 30,000-word sample.

After the tabulation had been made, dictionaries and word studies compiled by other groups were consulted for other words which are likely to be a part of scientific general language but which did not occur in the 30,000-word sample. These words were then added to the original tabulation.

Finally the tabulation, or lexicon, was expanded by supplementing all relevant inflectional forms of the paradigm families[2] represented by the words in the lexicon provided that the form is a part of the contemporary Russian language. After the expansion was completed, the lexicon contained approximately 170,000 entries, each entry consisting of a single Russian word (or idiomatic sequence of words) together with all essential English alternatives of that Russian word. For the details of the content and preparation of this lexicon, the reader is referred to the discussion by Micklesen.[3]

As indicated in the above discussion it is obvious that the main effort of the Machine Translation (MT) Project at the University of Washington has been directed toward the preparation of this lexicon. Since a tremendous amount of lexical material was available, the theoretical investigations were directed toward research problems which required such material. Solution of the problem of multiple meaning, for instance, requires a very careful study of words and their meaning. As a consequence, the problem of multiple meaning has been studied in detail at Washington. Studies in stem-ending dissection,[4] on the other hand, do not require such a lexicon and hence have not been carried out.

[1] Reifler, Erwin, Outline of the Project, LINGUISTIC AND ENGINEERING STUDIES IN AUTOMATIC LANGUAGE TRANSLATION. Report No. RADC-TN-58-321, ASTIA Document No. AD-148992, 1958.

[2] Paradigm family--One or more paradigmatic forms of the same "semantic unit"; for example: матрицы, матрице, матрицу are all formed from the stem матриц and hence they all belong to the same paradigm family.

[3] Micklesen, Lew R., Procedural Report, op. cit. in footnote 1.

[4] Stem-ending dissection is quite popular among MT research groups. At the time this report is being written, thirteen machine translation projects exist in the United States. All but two of these projects are concerned with Russian to English translation and, with the sole exception of the UW project, all are using, or intend to use, stem-ending dissection. Anthony G. Oettinger's scheme is probably the most complete (cf. his A Study for the Design of an Automatic Dictionary, Doctoral Thesis, Harvard University, 1954). It is quite probable that many different ways exis+ for accomplishing stem-ending dissection. The whole purpose of stem-ending dissection is to compress the dictionary by storing only the stem forms of Russian words

In its present form, the lexicon which has been compiled by the main project is intended for word-for-word translation. Several examples of translations are presented in Chapter 3; here it is just stated that such translations are far from clear and far from being readily intelligible. The work of the author has been twofold: first, to investigate computer processing methods for improving the quality of translations over that obtained by word-for-word techniques, and secondly, to evaluate the entire translation process with the objective of obtaining a tentative optimum design for a machine for translating languages. A brief discussion of the contributions of this thesis to these two problems is presented in the next two sections.

Computer Processing Methods for Improving the Translation

An extensive investigation was made of computer processing methods for the purpose of improving the quality of the translation. This work was done by Micklesen, Niehaus, and the author. Specific statements listing the contributions of each are reserved for the section where that particular problem is treated in detail.

The first step in the investigation was the preparation of a translation lexicon from the lexicon which was compiled by the main project. In this translation lexicon each individual entry contains certain grammatical and non-grammatical information which is necessary for logical processing. Computer programs which would resolve certain problems of multiple meaning for the University's IBM 650 computer were then written. The details of these programming techniques and examples of the results of the processing are presented in Chapters 4 and 5.

Design of a Special-Purpose Machine for Language Translation

The goal of machine translation research is the production of a machine which will produce accurately intelligible translations rapidly and economically. All investigations are directed toward furnishing data which may be used to attain this goal.

Chapters 6 through 9 of this report are concerned with an evaluation of the entire translation process and specification of a tentative design for a special-purpose digital-data processing machine which will provide the required properties of an efficient translating machine.

This work was done solely by the author.

---

and their inflectional endings, and by using computer programs to perform dissection and subsequent dictionary look-up on the incoming text words. Stem-ending dissection has two shortcomings: firstly, it is possible that some information may be lost in this technique, and, secondly, the processing programs are considerably more involved than for the case where all inflectional forms are entered in the dictionary.

Chapter 1

Statistical Aspects of Scientific Russian

In this chapter some of the statistical aspects of Scientific Russian will be considered. The purpose of this statistical study is to determine two things: first, whether the dictionary size would be smaller for the translation of scientific material than it would be for translation of general-language material; and second, whether it might be possible to predict an ultimate dictionary size, given only a limited text for analysis. The first of these two problems will be studied through a comparison of the scientific-language word count compiled by the linguistic section of the project with a general-language word count compiled by Josselson and others.[5] The second problem will be studied in consideration of the controversial hypothesis known as Zipf's Law.[6]

This study utilizes a list prepared from the scientific text sample analyzed by the University of Washington MT Research Group (UW),[7] and a general-language list compiled by Josselson.[6] These two lists were compiled on different bases since the UW list counts the frequency of every paradigmatic form,[2] while the Josselson list is essentially based on paradigm families,[2] for all form classes except pronouns, whose paradigmatic forms are counted individually. For this analysis the UW list was modified in order to make its rules of compilation identical with those of Josselson's list. The words of highest frequency in the two lists are presented in Tables A and B.

The word counts will be analyzed in two different ways: first, the probabilities of individual semantic units[8] in scientific-language and general-language word counts will be considered, and second, an analysis of the statistical properties of sets of words are considered, in which the members of a particular set are those semantic units which occurred in the text sample the same number of times. Thus all semantic units which occurred only once are in one set and those which occurred twice are in another set.

Probabilities of Individual Semantic Units

Table A (at the end of this chapter), taken from Josselson, contains a list of all Russian general-language words which occurred thirty-five or more times in a text sample of 29,345 words from Pushkin's "The Captain's Daughter." Table B (ibid.) lists the Russian scientific-language semantic units which occurred thirty-five or more times in the 30,000-word text sample analyzed by the UW project. The general-language list contains 102 words, and the scientific-language list 103 words. The lists are of almost the same length, but are by no means identical; 52 of the words in Table A did not occur at all in the entire scientific sample; and of the fifty-one which did occur, only thirty-eight are in Table B.

For words of highest frequency, the probability increases of scientific over general-language, and general-language over scientific material are listed in Tables C and D. In Table C the words are classified according to form class, while in Table D the words are listed according to frequency of occurrence. These data show clearly that for words of high probability a considerable difference exists between the usage in general language and usage in scientific discourse.

Statistical Characteristics of Low-Frequency Word Sets

Up to this point in the analysis only the probabilities of individual words have been considered. The emphasis will now be on the relationship between sets of words where each set consists of all words occurring in a sample the same number of times. For example, in the scientific-language sample, the set of all words occurring exactly once has a membership of 8197, while the set of all words occurring exactly 1000 times has a membership of zero.

The following discussion will show that only the sets with large memberships form consistent patterns.

[5] Josselson, Harry H., THE RUSSIAN WORD COUNT, Wayne University Press, 1953, pp. 23-30.

[6] Zipf, George Kingsley, HUMAN BEHAVIOR AND THE PRINCIPLE OF LEAST EFFORT, Addison Wesley Press, 1949.

[7] Micklesen, Lew R., op. cit., pp. 22-35.

[8] Semantic unit--Any meaningful portion of the source text selected to head a dictionary entry. It will sometimes correspond to a single free form, while at other times it will consist of more than one free form (idiomatic sequence) or less than one free form (constituent of a compound or complex). The term "semantic unit" was coined and defined by Reifler specifically for the requirements of MT (cf. his Some New MT Terms, op. cit. in footnote 1).

Since the sets with large membership contain only low-frequency words, the following investigation is concerned with the characteristics of low-frequency words.

A plot of the occurrences of words versus their rank order, (ordered according to the number of occurrences) with different word counts as the parameters, is presented in Figure 1. For words of low rank order the points show considerable randomness. One interesting characteristic is evident in this range: All samples from Pushkin show a characteristic upward concavity, while the sample from scientific Russian actually shows a characteristic downward concavity. These characteristics indicate that scientific Russian contains fewer low-rank-order words than does general-language Russian.

The total number of occurrences of words in the general-language high-frequency list is 13,755. Since the total word count is 29,345 words, the high-frequency list of 102 words accounts for about 47% of the total sample. The scientific 103-word high-frequency list accounts for 9124 of the 30,000-word total of the scientific sample, or about 30.4%. Again fewer occurrences of scientific high-frequency words are indicated than for general-language high-frequency words.

The fact that far fewer of the members of the scientific list are high-frequency words (9124 versus 13,755) indicates a considerably larger total vocabulary in the scientific sample. This does not, of course, indicate that an author in the field of general literature has a smaller vocabulary than an author in a highly specialized field. The scientific 30,000-word sample was compiled from the publications of a large number of authors in 40 different fields of science, while the general language sample was from only one author (Pushkin). Certainly it is borne out, however, that scientific discourse uses a vocabulary of considerable size, and hence a small translation lexicon is certain to yield unsatisfactory translations.

Zipf's Law

Zipf studied the statistical aspects of word frequencies, and on the basis of considerable empirical work, he postulated what is known as "Zipf's Law."[9] According to Zipf's Law, if the number of word occurrences in a sample of text are plotted on log-log paper against their rank order, then the points will lie roughly on a straight line. Mandelbrot has made a theoretical investigation of Zipf's "principle of least effort" using the analytical theory of thermodynamics and has stated that, instead of the straight-line relationship proposed by Zipf,actually two straight lines should be postulated for the plots: One line of negative slope for words of lowest rank order and another line of more negative slope for words of highest rank order.[10] Figure 1 shows that Mandelbrot's theory seems to hold true for the larger samples from Pushkin.

Zipf's Law cannot hold true for all sample sizes. The most frequent words such as conjunctions, particles, and pronouns will be used by the same author in any discourse with about the same probability of occurrence. Thus, if the size of the sample is increased ten times, the plot of such high-probability words will be moved upwards one cycle on the log-log paper. If Zipf's Law were to hold true for the entire rank order, the total number of words used by an author would increase proportionately with the size of the text. The number of words any person has at his command, however, is limited, and thus the straight-line relationship could not hold true for extremely large samples. Zipf rationalizes this by defining an "optimum sample size" which, in effect, is the largest sample for which a straight-line relationship will hold true.

The scientific sample does not show any appreciable deviation from a straight line, but the larger samples from Pushkin do show such a deviation.

Figure 1 shows several interesting characteristics. If a line is passed through the highest rank order points of each of the sets of words with a frequency of ten or less, such as through points A, B, C, D, E, F, it will be seen that the line is essentially straight in all cases. This is especially true in the case of the 29,345-word sample from Pushkin.

A complete membership list of the high-rank-order sets was not compiled from the scientific material. A count of words with a total occurrence of one was, however, made. They were found to total 8197. If a perpendicular is dropped to the rank order 1 abscissa from the point of intersection of this line with the rank order 2 abscissa, the perpendicular intersects the rank order 1 abscissa at about 5250. Thus (13,500-5250) = 8250 rank orders occur between the point of intersection of the averaging line with the perpendicular. This checks very closely with the count of 8197 words with occurrence of one in the scientific sample.

The surprising regularity of these sets for the larger sample sizes would seem to verify Zipf's and Mandelbrot's postulation that there is an underlying fundamental statistical law governing word frequencies which is independent of the meanings carried by the individual words. A relationship might be established which would indicate the size of vocabulary at the disposal of the author. For instance, in the case of the two 5,000-word samples and the 10,000-word sample from Pushkin plotted in Figure 1, the slopes of the lines through the highest rank order points in the low-occurrence sets are about the same. The 29,345-word sample, however, shows a considerably greater slope. Perhaps there is a relationship between the slopes of these lines through the high-rank-order points and the vocabulary size of the author. If so, this could allow an estimate of the vocabulary at the author's disposal, perhaps on the basis of a very small sample.

A careful statistical investigation of word frequencies might yield considerable information. Some possibilities are suggested in the next section.

[9] Zipf, George Kingsley, op. cit., p. 131.

[10] Mandelbrot, Benoit, An Informational Theory of the Statistical Structure of Language, COMMUNICATION THEORY, Edited by Willis Jackson, Academic Press, 1953, p. 486.

Further Work

The preceding analysis suggests several interesting investigations:

1. The sampling of another work of Pushkin in the same way, preferably a work with subject matter as different as possible from "The Captain's Daughter." Will this sample indicate the same rank-order-to-slope relationship?

2. Combination of the word count from "The Captain's Daughter" with that of another work by the same author. Is there a consistent relationship between slopes?

3. An exhaustive word count of as much of Pushkin's work as possible.

4. An exhaustive count of the work of another author in the field of general literature. How do the two counts compare with one another? Can we establish a criterion for vocabulary size?

5. A word count in individual fields of scientific endeavor. Can we establish criteria for vocabulary size in these fields? In particular, can a considerable saving in dictionary size be realized by restricting the lexicon to one field of science, or in other words, would the use of a micro-glossary be practical?

If a criterion for vocabulary size could be established, it would be invaluable for MT research; however this research was not attempted since the task would be a major one and funds and facilities were not available.

Conclusions

There is no indication that general-language discourse uses a larger vocabulary than scientific discourse. Three indications of this fact were presented in this chapter.

1. The high-frequency list in the scientific sample only accounts for 30.4% of the total number of word occurrences in the sample, while the general-language high-frequency list accounts for 47% of the general-language word occurrences. If the curve of occurrences versus rank order has any significance as Zipf's law insists that it has, then a much larger word list is indicated in the scientific sample.

2. The total paradigm-family count is 4775 for the general-language and 5137 for the scientific list. Again a larger scientific vocabulary is indicated.

3. The slope of the high-rank-order asymptotic in Figure 1 is greater for the general-language count than for the scientific word count for the two 30,000-word samples. Again a smaller general-language vocabulary is indicated.

TABLE A

High-Frequency General Language List (From Pushkin)

	Number of Occurrences		Number of Occurrences		Number of Occurrences
и	1160	себя	98	такой	41
я	777	крепость	97	видеть	40
в	724	так	96	другой	40
не	582	бог	81	пойти	40
что	484	стать	80	при	40
с	479	один	78	казак	38
быть	430	ли	75	казаться	38
на	421	комендант	73	когда	38
он	379	наш	73	старик	38
мой	297	ни	72	стоять	38
меня	297	батюшка	71	узнать	38
его	282	их	71	кто	37
ты	257	тебя	71	матушка	37
мне	256	уже	69	ну	37
к	244	бы	68	чтобы	37
сказать	236	ваш	63	голова	36
мы	220	еще	63	для	36
а	214	рука	63	сам	36
за	187	ей	62	тот	36
весь	184	дело	65	вдруг	35
то	180	увидеть	55	вас	35
свой	178	вы	53	злодей	35
этот	168	слово	53	лошадь	35
как	166	знать	51	письмо	35
о	137	минута	51	самый	35
который	134	твой	50	тут	35
ее	131	под	49		
но	131	говорить	48		
у	124	несколько	48		
из	123	человек	48		
по	122	спросить	47		
да	120	хотеть	47		
от	119	до	44		
она	111	отец	43		
же	109	дать	42		
отвечать	107	время	41		
ему	101	где	41		
мочь	98	какой	41		

TABLE B

High-Frequency Scientific Word List

	Number of Occurrences		Number of Occurrences		Number of Occurrences
и	1132	о	70	почва	44
в	1125	такой	69	еще	44
на	470	следующий	67	скорость	43
с	336	более	66	клетка	43
при	308	но	65	два	42
этот	276	его	65	растение	42
не	248	работа	62	путь	42
от	231	движение	59	воздух	41
для	220	действие	59	тело	41
который	210	только	58	часть	41
что	207	между	58	поэтому	41
по	203	ее	57	после	40
быть	196	если	56	образовать	40
или	185	так	56	газ	40
из	178	также	56	некоторый	40
к	174	различный	56	ошибка	39
тот	173	основной	55	место	39
а	139	за	52	группа	39
весь	123	мы	51	самолет	39
как	113	все	51	например	39
мочь	113	пункция	51	же	38
их	110	обычный	51	температура	38
большой	106	высокий	51	через	37
до	102	величина	51	них	36
иметь	101	форма	50	конец	36
у	99	значительный	50	полный	36
случай	93	первый	48	чем	35
являться	90	они	47	масса	35
один	89	необходимый	47	точка	35
время	83	свой	46	частица	35
система	80	должен	45	сторона	35
дать	75	вес	45	общий	35
поверхность	74	строй	45		
вид	72	под	45		
другой	71	несколько	44		
вода	71				

Table C

Occurrences of High Probability Words

According to Form Class

Personal Pronouns

		Scientific	General Language	Increase GL/SC.
я	(I)	16	777	3360
он	(he/it)	21	379	3580
она	(she/it)	14	110	522
вы	(you)	0	53	U
мой	(my/wash)	0	297	U
меня	(me)	0	288	U
тебя	(you)	0	71	U
ему	(him/it)	0	101	U
ваш	(your)	0	63	U
свой	(your)	0	178	2070
твой	(yours)	0	50	U

Adjectives

	Scientific	General Language	Increase GL/SC.
следующий (following)	67	0	U
различных (different)	56	0	U
основной (basic)	55	0	U
большой (big)	106	0	U

Subordinating Conjunction

	Scientific	General Language	Increase GL/SC.
чтобы (in-order/that)			

Interrogative Indefinite Adjective

	Scientific	General Language
какой (what/which/any)		41

Substantives

	Scientific	General Language
бог (god)	0	81
крепость (strength,fortress)	0	97
комендант (commander)	0	73
батюшка (father)	0	71
рука (hand)	0	63
слово (word)	0	63
человек (man/person/(of)people)	0	48
отец (father)	0	43
казак (Cossack)	0	38

Substantives (continued)

	Scientific	General Language
матушка (mother)	0	37
голова (head)	0	36
злодей (villain)	0	35
лошадь (horse)	0	35
письмо (letter/(hand)writing)	0	35
случай (case/chance/occurrence)	93	25
вода (water)	71	25
работа (work)	62	25
движение (motion/movement)	59	25
действие (act/action/effect/operation)	59	25

Adverbs

	Scientific	General Language
вдруг (suddenly)	0	35
когда (when)		38
тут (here/there)		35
более (more)	66	0

Pro-Substantives

	Scientific	General Language
сам (self)		36
кто (who)		37

Pro-Adjectives

	Scientific	General Language
другой (other)		40
такой (such)		41

Table C (continued)

Multiple Form Class

	Scientific	General Language
Verb-Substantive		
стать (to become/begin/ stand/form/state)		80
Particle-Predicate Adjective-Adverb		
уже (already//(is/are)more-narrow(ly))		69
Particle-Conjunction		
да (yes/let/but)		120

Verbs

	Scientific	General Language	Increase GL/SC.
знать (to know)	0	51	
спросить (to ask)	0	47	
хотеть (to want)	0	47	
пойти (to go(off)/take-after)	0	40	
казаться (to seem)	0	38	
стоять (to stand)	0	38	
узнать (to learn/recognize)	0	38	
иметь (to have)	101	0	U
являться (to be/appear)	90	0	U

Prepositions

	Scientific	General Language	Increase GL/SC.
при (at/with/before/ in-time-of)	308	40	450
для (for)	220	36	420
в (in/at/to/on/of/like)	1125	724	37
от (from/of/for)	231	119	58
по (on/by/along/for/in)	203	122	33
из (out/from)	178	123	12
до (up-to/before)	102	44	69
между (between/among)	58		U

Intensive Adjective

	Scientific	General Language
самый (the-very/most)		35

Interjection

	Scientific	General Language
ну (well)		37

Table D

Comparison of Probabilities of General Language and Scientific High Frequency Word Lists

General Language	Increases in Probability	Scientific List	Increase in Probability of Scientific over General Language Occurrence
и	0	и	0
я	+3360	в	+37
в	25	на	0
не	+108	с	-16.0
что	+104	при	+450
с	+23	этот	±35.7
быть	+106	что	-44.7
на	0	не	-46.5
он	+3580	от	+58
мой	u	для	+420
меня	u	который	+23
его	+272	по	+32.6
ты	u	быть	-48.4
мне	+7460	или	u
к	±13.8	из	+12.2
сказать	+2470	к	-9.8
мы	+154	тот	0
а	+15.1	а	-10.3
за	+195	весь	-11.2
весь	+16.1	как	-8.7
то	0	мочь	0
свой	+2070	их	+11.4
это	-21.7	большой	u
как	+12.7	до	+69.4
о	+60	иметь	u
который	-15.7	у	0
ее	+86.5	случай	u
но	+52	являться	u
у	0	один	0
из	-8.5	время	+40
по	-19.7	система	u
да	u	дать	+31
от	-30	поверхность	u
она	+522	вид	u
же	+114	другой	+17
отвечать	+2070	вода	u
ему	u	о	-25
мочь	0	такой	+8.8
себя	+79	следующий	u
крепость	u	более	u
так	±26.1	но	-25.7
бог	u	его	-63
стать	u	работа	u
один	0	движение	u
ли	+1020	действие	u
комендант	u	между	u
наш	+114	ее	-35.7
ни	+576	если	u
батюшка	u	так	-15.6
их	-7.4	также	u
тебя	u	различный	u
уже	u	основной	u
бы	-275	за	-54
ваш	u	мы	-54.5
еще	u	все	u
рука	u		
ей	+1750		

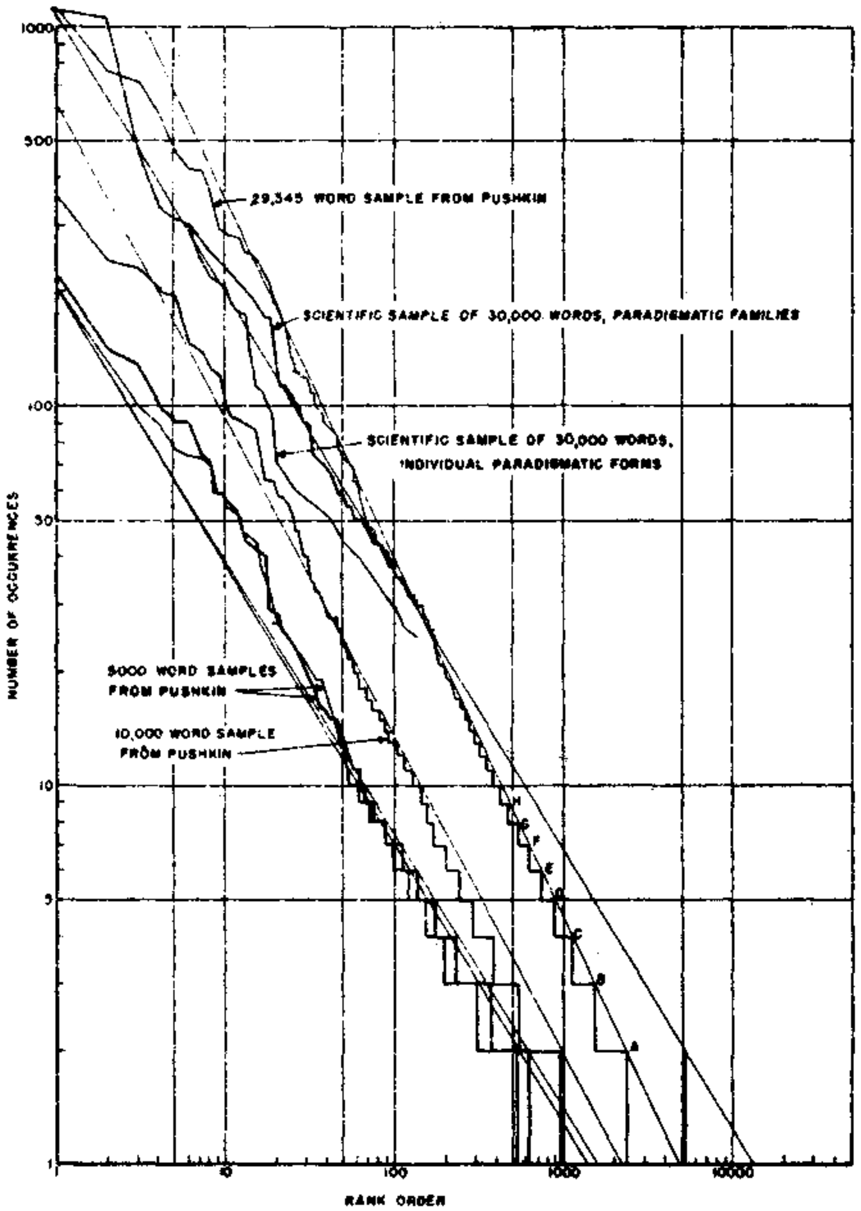


FIGURE I. RANK ORDER VERSUS FREQUENCY DISTRIBUTION FOR VARIOUS WORD COUNTS

Chapter 2

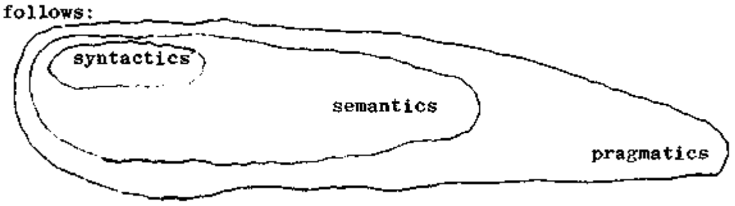
The Semantic Theory of Information and Machine Translation

In this chapter some of the philosophical aspects of the human communication problem will be considered, together with the constraints imposed by digital data-processing equipment upon the processing of different sorts of communications. Such a discussion is important before the details of the computer programming operation are presented in order that these programs may be viewed in their proper perspective.

Semiotic Theory

The whole broad study of language and sign systems is called semiotic, and is studied at three different levels representing different degrees of abstracting: syntactics (the study of signs and the relations between signs), semantics (the study of the relationship between signs and their designata), and pragmatics (the study of signs in relation to their users). A simple example of a syntactic relationship would be that of the letters "q" and "u" in printed English; the letter "q" is almost always succeeded by "u". An example of a semantic relationship would be that between a certain four-legged animal and the word "dog." Finally an example of a pragmatic relationship would be that between the words: "come here" and the response of the listener, or the effect of such emotionally charged words as "nigger," "scab," and "chippy" on a listener.

These three studies, syntactics, semantics, and pragmatics do not exist as separate and exclusive entities but overlap as follows:



or: (syntactics) $\subset$ (semantics) $\subset$ (pragmatics)

Pragmatics

The largest of the three areas, pragmatics, includes both of the other two areas. Pragmatics has been extensively studied in the field of experimental psychology, but the studies are essentially qualitative and empirical rather than theoretical as in the case of the Shannon-Wiener development in the area of syntactics.[11, 12] Such studies as word-association tests are typical of psychological investigations into pragmatics.

In the broadest sense, MT is concerned with pragmatics, since ideally (at least for scientific texts) the translated material should have the same effect on a reader who is a native speaker in the target language as the original material would have on a native speaker of the source language. A capability of giving translations which are correct in a pragmatic sense would allow translation of poetry. At the present time, however, research groups are only hoping for a translation which is correct according to the semantics and syntactics of language, and acceptable translations of poetry are generally considered as too ambitious to be considered.

There are two reasons why pragmatic problems in the narrow sense of the term are not considered in contemporary MT research. First, pragmatics is concerned with conditioned, or emotional responses to words. Since scientific language is objective, such problems are not likely to occur. Second, solutions to pragmatic problems are very difficult or even impossible to include in MT processing programs. Examples of difficult pragmatic problems are easy to concoct. Ivanov gives several examples from poetry.[13] One interesting example

[11] Shannon, Claude E., and Warren Weaver; THE MATHEMATICAL THEORY OF COMMUNICATION, University of Illinois Press, 1949.

[12] Wiener, Norbert; CYBERNETICS, John Wiley and Sons, 1948.

[13] Ivanov, V. V., Linguistic Problems Connected with Poetry Translation, ABSTRACTS OF THE CONFERENCE ON MACHINE TRANSLATION, May 15-21, 1958. U. S. Joint Publications Research Service, JPRS/DC-241, July 22, 1958, p. 24.

of a pragmatic problem occurs in the conventions regarding the usage of the first person pronoun in Czech and Russian. The first person pronoun is used conventionally with its appropriate verbal forms in Russian without any stigma being attached to the style. In Czech, on the other hand, the use of the first person pronoun is restricted to those situations where emphasis is desired. If a Russian article is translated into Czech with the first person pronouns retained, then a translation may result which will seem to a Czech as boastful and egocentric in the extreme. It is therefore quite possible to translate from Russian into Czech with complete adherence to the syntactic and semantic requirements of the two languages, and yet obtain a translation that fails to convey the message desired by the author, even though the message was well expressed in the original article.

Pragmatic problems are not considered in this thesis; they are mentioned merely to point out some of the more general problems in MT.

Semantics

Semantic theory is concerned with "meaning." Cherry points out that the word "meaning" must be carefully defined since the word "meaning" may be interpreted in two different ways: First, there is the idea of equivalence connected with "meaning"; are two statements logically true, i.e., are the two statements equivalent? The second interpretation is that of extra-linguistic truth and reference, or whether a statement is true "in fact" and experience. As an illustration, Cherry uses the example of replacing the word "brine" for "salt water" in the sentence, "Salt water is a good emetic," to give, "Brine is a good emetic." These two sentences are certainly equivalent; they have the same "meaning" in the first sense. As to whether they are also true according to fact and experience could only be determined by an unattractive empirical approach to the problem.[14] For the purposes of the MT researcher, only the idea of equivalence is of interest. In MT the assumption is made that the original author wrote material which is true according to experience. Machine Translation is only concerned with obtaining an equivalent which "means the same thing" in another language.

The meaning problem in Machine Translation is one of "hypothesis" and "evidence." There is a hypothesis that the target language[15] equivalent of the source language word[16] "матрица" is "matrix" (and not "die"), or that the equivalent of "школы" is "of school" (and not "schools"). In a certain context there will be evidence which either supports or contradicts these hypotheses. The problem is to determine whether the evidence warrants selecting one equivalent over the other. This problem must be solved by searching the context of the word for pre-determined words or word patterns. Some computer routines used for context analysis will be used very often during translation, others will be used only rarely. Often these routines for context analysis will only increase the probability of one particular equivalent and reduce the probability of others. It can be demonstrated that mere selection of the most probable translation on the basis of classical grammar may lead to an incorrect decision. Selection of the "most probable" translation is not enough; rather <u>certainty</u> is what is required.

Source-language words frequently have a multiplicity of meanings, all of which are not embodied in one target-language word. Such source-language words are then equivalent to several target-language words, at least, as many as will translate the intended meaning of the source-language word in all contextual environments. An example of this would be that of the Russian adjective, шальной which in one context will convey the meaning of the English "foolish" while in another the meaning of "random." In other instances, particularly in the area of grammatical categories, exact equivalents are very difficult to find. For instance, in Russian there is the grammatical category of aspect, the main subdivisions of which are termed the perfective and imperfective aspects. The perfective aspect expresses the absolute, single completion of the action of the verb while the imperfective does not. The perfective and imperfective aspects of the Russian verb do not have counterparts in the English language; therefore the exact connotations of these aspects very seldom, if ever, can be captured in the English language.

The ideas expressed by some Russian reflexives are also extremely difficult to translate. For instance, мне нравится would probably be translated as "I like" or "it is pleasing to me." Neither is an accurate translation, however, since both convey the idea of continuing affection while "нравится" has the connotation of first impression. Also both "like" and "pleasing" are emotionally a little too mild since "нравится" is somewhere between "like" and "love." In instances such as these, regardless of how much processing is done, more than one equally likely English equivalent will remain.

The semantic problems must be solved by the linguists and programmers since, as will be discussed in the next section, the translating machine must operate entirely in the realm of syntactics.

Syntactics

The smallest area, syntactics, represents the only study of the three for which an extensive mathematical theory has been developed. This area represents the field which has the mathematical model known as the "statistical theory of communication," advanced originally by Shannon and Wiener.[17,18] Sometimes the Shannon-

[14] Cherry, Colin; ON HUMAN COMMUNICATION, John Wiley and Sons, 1957, p. 112.

[15] Target language--The language into which the translation is made.

[16] Source language--The language from which the translation is made.

[17] Shannon, Claude E., op. cit.

[18] Wiener, Norbert, op. cit.

Wiener theory is called "information theory," but this is actually a misnomer. The Shannon-Wiener theory is more accurately a theory of the information potential of a sign system; it does not attempt to define what the signs may or may not designate. The statistical theory of communication is, therefore, entirely a syntactic theory.

As was pointed out at the end of the last section, syntactics is the fundamental area of work in MT. Digital data-processing equipment must necessarily work entirely in the area of syntactics since the machine can only interrogate various bit positions of the data for patterns which it is programmed to recognize. Any semantic or pragmatic characteristics of language which are to be considered in the translation process must be specified by syntactic relations before they can be programmed. The semantic and pragmatic problems of MT, then, are necessarily solved by humans before the translation programs are written.

The following chapters, therefore, use concepts which are entirely abstracted from semantics and pragmatics as such. In the next chapter, for example, a criterion for measure of quality of translation is developed. This is usually considered to be a problem in judgment, or subjective in nature. By reducing the problem to pure syntactics, the problem becomes objective, and as will be shown, gives a result which is compatible with the intuitive concept of translation quality, at least as far as multiple meaning is concerned.

Chapter 3

Translation Quality

In this chapter the nature of translations will be considered, along with criteria for evaluating these translations. Fundamental to evaluating translations by machine is the requirement for some standard, against which their quality may be measured. In this chapter a measure of translation quality is developed.

The Nonuniqueness of the Translation

Translations by humans are by no means unique. The same Russian article would not be translated in the same way by two or more equally able human translators, nor would one and the same person be likely to translate an article in exactly the same way at different times. Since no two translations are exactly alike, no two will really express the same semantic message and hence the quality of each of these translations will be slightly different. The relative evaluation of such translations is extremely difficult to make, and invariably such an evaluation is as much subjective as it is objective. Sometimes an actual error will occur which is easily detectable, such as the case which the author encountered in a catalog from a bookstore in which ТЕОРИЯ МАТРИЦ was translated as "Theory of Dies," when it should have been "Theory of Matrices." On the other hand, the sentence она мне нравится might correctly be translated as "she is pleasing to me," but as was pointed out in Chapter 2, this translation does not convey the intended shade of meaning.

The Problem of Translation Quality

The problem of translation quality has been considered by Miller and Beebe-Center.[19] They list the following characteristics of a scale for measuring translation quality: "A scale of the quality of translations should be reliable, valid, objective and easy to use." By "reliable" they mean that the measure should be independent of time and of the person calculating the measure.

The criteria for quality of translation listed by Miller and Beebe-Center would seem to be sufficient for either human or machine translations. These criteria will, therefore, be used to evaluate possible measures discussed in the following sections.

Subjective or Human Evaluation of Translation Quality

The most obvious criterion for translation quality would be that of human evaluation, but such a procedure would violate the requirements of objectivity and reliability. Reliability could be improved by using a larger group of judges, but this violates the requirement "easy to use." The human, or subjective criterion for evaluation of translations does not seem to lend itself to procedures which will satisfy the characteristics of Miller and Beebe-Center, listed in the last section, and hence the subjective criterion is rejected.

The difficulties encountered in this section would seem to come about because too much is being attempted at once in these evaluations. No specific prior agreement is made between judges concerning just what is "good" (and how good) and what is "bad" (and how bad) in a translation. Since this report is concerned only with translations by machine, it is well to examine an example of a machine translation to determine whether some restrictions might be placed on the definition of translation quality in order to allow a precise, though restricted, measure. The basic machine translation is the word-for-word translation or that obtained by an in-isolation translation of each individual text word. All processed machine translations are obtained from the word-for-word translation.

The Word-for-Word Translation

The word-for-word translation is obtained when each individual word[20] is translated in isolation. The equivalents in the language ipto which the translation is made must include every unique usage for any context of that original word which is being translated. For example, consider the Russian phrase: О лечении нервной импотенции новокаином. The word-for-word translation of this phrase as extracted from the MT-operational lexicon is as follows:

about/against/with   treatment   (of)(to/for)(by/with/as) nerve/nervous   (of)(to/for)impotence(s)

[19] Miller, George and Beebe-Center, J. G., MECHANICAL TRANSLATION, Vol. 3, No. 3, December 1956, p. 73.

[20] Sometimes also a small group of words (idioms) is treated as an individual word.

(by/with/as)novocain

Some remarks concerning the conventions of the notation are in order. If one or more alternatives are enclosed in parentheses, the reader has a choice of using or not using one of these alternatives. If alternatives are not enclosed by parentheses, one of them must be chosen. For instance, since "about/against/wīth," the English equivalent for the Russian preposition "o", is not within parentheses, one of the three alternatives, "about," "against," or "with" must be chosen. Thus there are three possibilities. In the case of the equivalent of нервной namely "(of)(to/for)(by/with/as) nerve/nervous," the reader may choose either any one of the six English prepositions, or none of them, and either "nerve" or "nervous." Consequently, there are fourteen possible combinations.

Several phrases could be formed from the word-for-word translation; three possible translations which would probably occur to a human reader are as follows:

(1) About treatment of nervous impotence by novocain.

(2) About treatment for nervous impotence with novocain.

or a more free translation,

(3) The use of novocain for the treatment of nervous impotence.

The first two translations may be obtained from the word-for-word translation by merely ignoring unwanted equivalents. Word order is in both cases the same as in the word-for-word translation. The third translation attempts much more: a word-order rearrangement is made and also the preposition "for" is used which is not considered in the operational lexicon as equivalent to the Russian preposition "o." These examples illustrate the fact that solution to the multiple meaning problem generally gives translations which are accurately intelligible but that multiple meaning is certainly not the only problem. A complete definition of translation quality should consider <u>three</u> problems: multiple meaning, word order, and changes in structure of the sentence during translation, since sometimes all three of these techniques must be used to obtain a completely idiomatic translation.

In this report only the problem of multiple meaning will be considered in detail, and hence the definition of translation quality will be elaborated to measure only this one part of the translation problem.

In order to develop a definition of the measure of translation quality, some of the concepts of statistical communication theory are helpful. These concepts are presented in the next section along with discussions concerning the relationships between problems of the measure of translation quality and those problems usually considered in classical communication theory.

<u>Information as Defined in Statistical Communication Theory</u>

Statistical communication theory is concerned with the transmission of signals, with no regard for what the signals may or may not represent. Statistical communication theory, therefore, is not an "information theory" since it is not concerned with semantics in any way. Rather statistical communication theory is concerned with an "information potential" of a system of signals. This theory defines "information received" by the following equation:

$$\text{Information received} = \log_2 \frac{\text{probability at the receiver of the event after the message is received}}{\text{probability at the receiver of the event before the message is received}} = \log \frac{p(B)}{p(n,n-1,n-2,\ldots)}$$

Where: p(B) = probability of the event after receiving the message

p(n,n-1,n-2,...) = probability of the event before sending the message, i.e., the probability of the nth message given the $(n-1)^{st}$, (n-2) etc., messages.

As an example, if the signals are letters of the English alphabet, then at a particular instant the probability of the next letter being "e" is .131. Hence the information received if the next letter does turn out to be "e" is:

$$I_{\pi} = \log \frac{1}{.131} = 2.93$$

In statistical communication theory the definition of information content does not concern itself with the semantic content of a message; the message may consist of pure nonsense and yet the "information received" may be considerable. This divorce from semantics is necessary if subjectiveness is to be avoided. Mathematical logic, communication theory and any form of computer processing must necessarily operate entirely in the realm of syntactics. A mathematical theory probably cannot give any solutions to semantic problems; hence the measure of quality of translation will also be divorced from semantics, and like communication theory, will be based entirely on syntactic relationships.

A measure of translation quality may be defined which will satisfy the following requirements:

1. In accordance with the definitions of statistical communication theory, measure of the MT product should be a logarithmic function;[21] i.e., the measure should give an indication of the logarithm of the number of sequences which can still be derived from the translation.
2. The measure should give a quality of "0" for the crude word-for-word translation; and a measure of "1"

[21] The reasoning behind the advisability of using a logarithmic function may be found in many good books on information theory. For an especially good discussion see Cherry, op. cit., p. 178.

when all the multiple meaning problems have been solved, and consequently, only one possible sequence exists.

The translation quality for an "n" word sequence may now be defined by the following equation:

$$T = \frac{\log_2(p_1 p') + \log_2(p_2/p'_2) + \dots + \log_2(p_n/p'_n)}{\log(p_1) + \log(p_2) + \dots + \log_2(p_n)} = \frac{\log_2(\pi p_i)/(p'_i)}{\log_2(\pi p_i)} \tag{2}$$

where: $p_i = 1/s_i$, s_i = number of possible English alternatives in the word-for-word translation for the ith semantic unit of the source language.

$p'_i = 1/s'_i$, s'_i = number of possible English alternatives in the processed translation for the ith semantic unit of the source language.

We may also define the information contribution of a particular processing routine to a particular translation of the above definition:

$$I = T_{output} - T_{input} = \frac{\log_2(\pi p_i)/(\pi p'_{i'})}{\log_2(\pi p_1)} \tag{3}$$

where: $p'_i = 1/s'_i$, s'_i = number of possible English alternatives in the input material to this particular processing for the 1th semantic unit of the source language.

$p'_i = 1/s^{\circ}_i$, s''_i = number of possible English alternatives in the output material from this particular processing routine for the ith semantic unit of the source language.

In the case where the input is the word-for-word translation, I equals T. It also follows that the remaining information required to solve the multiple meaning problem completely is:

$$I_r = 1 - T$$

Two examples of applications of these definitions will now be presented. First, suppose that a processed translation of О лечении нервной импотенции новокаином is as follows:

about/against/with treatment (of)(to/for)nerve/nervous impotence(s) (by/with/as)novocain

The measure of completeness of this processed translation is

$$T = \frac{\log_2(3/3)(1/1)(14/8)(8/2)(4/4)}{\log_2(3)(1)(14)(8)(4)} = \frac{2.81}{10.40} = .27$$

The second example is the following:

стенки сосудов пронизаны крупными диаметр 6-10μ, в более широких сосудах до 15μ, окаймленными, округлыми, овальными, или многоугольными порами с широким окаймлением и с чечевицеобразным горизонтально вытянутым, не заходящим за окаймление.

Word-for-word translation:

(of)wall(s) (of)vessels (are)perforated/threaded (by/with/as)large/important, diameter 6-10μ, in/to/at/on/of/like more (of)wide vessels (up)to/before 15μ, (by/with/as)edged, (by/with/as) round(ed), (by/with/as)oval or (by/with/as)polygonal (by/with/as)pores/times/seasons with/from/about (to/for)(by/with/as)wide (by/with/as)edging and/even/too/-- with/from/about (to/for) (by/with/as)lens-shaped (is)horizontal(ly) (to/for)(by/with/as)extracted/elongated/exhausted (by/with/as)opening/mesh, not (to/for)(by/with/as)going/dropping-in/setting behind/beyond/for/after/in/at/-- edging.

Processed translation:

(of)wall(s) (of)vessels (are)perforated/threaded (by/with/as)large/important, diameter 6-10μ, in/at more wide vessels (up)to/before 15μ, (by/with/as)edged, round(ed), oval, or (by/with/as)polygonal pores/times/seasons with wide bordering/flange/burr and/even/too/--- with lens-shaped horizontally extracted/elongated/exhausted opening/mesh not (to/for) (by/with/as)going/dropping-in/setting behind/beyond/for bordering/flange/burr.

The translation quality of this relatively long sentence may be calculated from Eq 3 as follows:

$$T = \frac{\log_2 \dfrac{(4)(2)(2)(8)(5)(2)(4)(4)(8)(4)(4)(12)(3)(8)(12)(2)(3)(4)(4)(12)(8)(4)(5)(3)}{(4)(2)(2)(8)(2)(4)(4)(2)(4)(3)(2)(3)(2)(3)(2)(4)(2)(3)}}{\log_2(4)(2)(2)(8)(5)(2)(4)(4)(8)(4)(4)(12)(3)(8)(12)(2)(3)(4)(4)(12)(8)(4)(5)(3)}$$

$$\frac{\log_2(2.95 \times 10^7)}{\log_2(5.01 \times 10^{15})} = \frac{24.8}{52.1} = .475$$

The value of the information required to solve the remaining ambiguities in the example is:

$$I_r = I - T = 1 - .27 = .73$$

For the second example it is:

$$I_r = 1 - T = .525$$

Conclusions

In this chapter a measure of quality of translation has been presented which gives a criterion which is reliable, objective, and easy to use. In addition, it would seem to be a valid indication of the difficulty which would be encountered by a human reader in accurately understanding the text.

The measure does not presume to evaluate the accuracy or correctness of the solution to the problem of multiple meaning; only the completeness of the solution is measured. In other words, if a program were written which would arbitrarily select the first alternative in every instance where a multiple meaning problem exists, the program would be credited with complete solution. In applying the measure, therefore, it must be assumed that the processing programs are valid.

The measure does not consider all factors of the general processing problems of machine translation. Word-order rearrangement and necessary changes in the structure of the sentence must sometimes be accomplished if idiomatic translations are to be obtained. These factors are of lesser importance than multiple meaning, and as yet have not been specified in sufficient detail to allow inclusion in the measure. When these factors have been specified accurately, undoubtedly a standard of measure can be developed for them.

The measure defined by Equation 2 will be used extensively in the following chapters to demonstrate quantitatively the effects of the processing routines on the translations.

Chapter 4

Precise Specification of the Processing Steps

Since word-for-word translation is not satisfactory for accurate intelligibility, the translation must be improved by the use of syntactical considerations. In machine translation, syntactical information provides the basis for designing computer processing programs. In this chapter and in Chapter 5, the processing programs developed for the investigations described in this report are discussed. This chapter is concerned with the linguistic specification of the processing steps and the way these specifications are used to develop the processing programs.

Solutions of the various problems encountered in machine translation require a detailed specification of the languages involved. The first step in the construction of computer algorithms for translation processing is therefore a careful analysis of text material by a linguist. In his analysis of that material, the linguist must determine just where ambiguities exist and just what properties of syntax will resolve these ambiguities. The following example will illustrate a typical intended meaning problem encountered in machine translation.

из	всех	представляющихся	способов	осуществления	эталона
out/from	(of)all	(of)(being)(re)presented	(of)methods	(of)realization(s)	(of)standard

выбираются	конечно	те	которые	позволяют	обеспечить
are-chosen/taken-out/get,	(is)final/finite(ly)/of-course,	those,	which	allow	(to)secure/provide

наибольшую	метрологическую	точность	воспроизведения	единицы
the-biggest	(metrological)	accuracy	(of)reproduction(s)	(of)unit/one(s).

Here the semantic unit of the source language is listed directly above its target-language equivalent. The features of syntax which specify the intended grammatical meanings of some of the semantic units in this sentence are as follows:

	Specification	Action
всех	pro-adjective in genitive or locative; preceding preposition governing the genitive case	eliminate English prepositions
представляющихся	present active participle; preceding agreeing adjective	eliminate English prepositions
способов	substantive; preceding agreeing adjective	eliminate English prepositions
осуществления	substantive in nominative-accusative plural or genitive singular; immediately preceding substantive suggests genitive singular	eliminate (s) retain (of)
эталона	substantive in genitive with preceding substantive	retain (of)
конечно	multiple distribution class (particle, predicate adjective, or adverb); with preceding and following commas the particle is indicated	eliminate (is) final/finite(ly) retain of-course
обеспечить	verb, infinitive, with preceding finite verb which requires "TO"	retain (to)
воспроизведения	substantive in nominative-accusative plural or genitive singular; preceding substantive suggests genitive singular	eliminate (s) retain (of)

единицы:	substantive in nominative-accusative plural or genitive singular; preceding substantive suggests genitive singular	eliminate (s) retain (of)

At first glance it would seem that the specifications indicated above are complete and adequate, but if one should attempt to write a computer algorithm which would automatically perform these interrogations and the indicated processing, it would soon become evident that the specifications are incomplete. More information is needed to completely specify the necessary and sufficient conditions for resolving the indicated problems of multiple meaning. For instance, it has not been specified just how far backward or forward in the text it may be necessary to search in order to establish whether a particular condition exists. Also, the permissible distributional classes of semantic units which may occur between the word being processed and the word being searched for have not been specified.

It is necessary that the linguist supply necessary and sufficient information about the requisites for the unique determination of intended meaning. If sufficient information is not provided, the program for the computer cannot be expected to perform in the desired manner in all cases. On the other hand, if more properties than those which are absolutely necessary are considered, the program will be more restricted and longer than necessary. To aid in the determination of completeness, a system of logical expressions has been developed. These expressions are described in the next section.

The Logical Expressions for Specification of Syntactical Properties

The logical expressions for the specification of syntactic properties are written in the form of an implication where the portion to the left of the implication sign refers to the source language syntax, and the portion to the right of the implication sign refers to the action to be taken regarding the translation if the properties denoted on the left hand side are found to exist.

The left-most expression in the implication specifies the grammatical and non-grammatical properties which must be met by the word which is being processed if the expression is to apply. The other expressions to the left of the implication sign indicate properties which must be satisfied by the environment of the word being processed if the expression is to apply. The expressions to the right of the implication sign indicate the action to be taken provided all conditions on the left-hand side of the implication sign are satisfied. First, the form of the individual expressions are presented, then several examples of the implications.

The Main Classifications and Superscript and Subscript Notation

The individual expressions used to denote properties of source-language semantic units are described in this section. The list is not exhaustive; only those definitions are made which are required for the processing described in this thesis. The general form, E, of the individual expressions is as follows:

$$E = X_a^b$$

Where X is the position of the main classification.

General Form of the Individual Expressions

The notation is described specifically below. The main classifications are as follows:

S -- substantive
2 -- prosubstantive
V -- verb
D -- adverb
E -- expression of some type (exact type indicated by superscript as described below)
R -- particle
C -- conjunction
K -- comma
N -- numeral
[] -- the entire implication applies uniquely to the word enclosed by the square brackets.

G -- gerund
A -- adjective
-- proadjective
U -- universal class -- may be any word
P -- preposition
B_1-- left-hand bracket
B_2-- right-hand bracket
{}-- allowed intervening words. If no allowed intervening words are indicated, only adverbs and particles (which may occur anywhere) are allowed.

The subscript is the position where certain grammatical information about the expression is denoted, such as gender, number, case and coordination. The superscript notation includes all necessary information not carried by the subscript.

For substantives, prosubstantives, adjectives, proadjectives, and prepositions, the notation is as follows:

Right-hand Subscript (position a)
N -- nominative case
G -- genitive case
D -- dative case

Right-hand Superscript (position b)
1. Substantives only
a -- animate
c -- concrete

A -- accusative case
I -- instrumental case
L -- prepositional
(locative) case
* -- used to denote case agreement
(see examples)

2. Adjectives only
r -- participle (adjectival)
a -- active
p -- passive
n -- present
t -- past

For case expressions an additional subscript is required for number:

s -- singular
pl -- plural
(i.e., genitive singular is expressed as G_s)

For conjunctions only two subscripts are required:

<u>Right-hand Subscript</u>
c -- coordinating
s -- subordinating

For verbs, the subscripts denote person and number:

<u>Right-hand Subscript</u> (position a)
1,2,3, -- person
s -- singular
pl -- plural

<u>Right-hand Superscript</u> (position b)
r -- reflexive
a -- active
p -- passive
I -- infinitive

The Operators

The operators denote the direction and distance over which the search, or logical operation, may proceed. They are defined as follows:

p_n -- denotes that <u>preceding</u> text is to be searched, and is to be searched <u>n</u> words back (or to the beginning of the sentence, whichever comes first) and no further. If no number is placed in the "n" position, the search may proceed backwards to the beginning of the sentence or until an unallowed word is encountered.

f_n -- same as p_n, except requires <u>forward</u> search (toward <u>end</u> of sentence).

The Connectives

The connectives are defined wherever possible to conform to the conventions of logical algebra:

. -- logical AND
+ -- logical OR
~ -- negation
⌒ -- requires (i.e., "V ⌒ to" means "verb requires English "to" in front of word being processed.")

The Right-Hand Side of the Implication

<u>Main Classification</u>
N -- nominative case chosen
G -- genitive " "
D -- dative " "
A -- accusative " "
I -- instrumental " "
L -- prepositional" "
(s) -- plural form required
(+ s) -- add plural form
P -- refers to English prepositions whose function is expressed in Russian by an inflection, such as English "of" being expressed in Russian by a genitive inflection. These English prepositions must be inserted in the translation as required.

<u>Subscript Notation</u>
.* -- take case intersection of agreeing expressions on left side of implication denoted by ".*"

Two examples of implications will be presented first, then the required implications are presented for the ambiguities of the sentence presented at the beginning of this chapter.

Example 1:

$$\text{Ꝟ}\cdot_{G} \cdot p \; P \cdot_{G} \longrightarrow \sim P$$

This implication conveys the fact that it applies only to pro-adjectives (Ꝟ) which have the inflectional form of the genitive case. The fact that logical AND (.) is placed before the symbol for the genitive case expresses the fact that this inflectional form may have the possibility of being some other case also, but it must have the possibility of being genitive. The "p" is an operator and denotes the fact that the processing program is to search backwards in the sentence being processed for a source language preposition (P) which may govern the genitive (and perhaps some other case as well). The fact that no subscript is placed on the "p" expression denotes that the search is to proceed backwards until either the required preposition is found, until the beginning of the sentence is reached, or until an unallowed intervening word is encountered, whichever one of the three conditions occurs first. The fact that no expression for an allowed intervening word occurs in the implication denotes the fact that only adverbs or particles (which may occur anywhere) are allowed to intervene.

The implication sign and the expressions to the right of the implication sign then denote the fact that if all the conditions expressed on the left-hand side are satisfied, then no (~) English preposition (p) is to be placed before the word being processed in the sentence.

A more general expression, which includes Example 1 as a special case, is presented below:

Example 2:

$$._{*} \cdot p^{P} ._{*} \longrightarrow \sim P$$

This example is identical to Example 1 except that instead of the subscripts ($._G$) which denote just that genitive possibilities are to be considered, this implication has the subscripts ($.*$) which denote that if the pro-adjective has any cases in common with a preceding preposition (separated only by adverbs or particles) then no English prepositions are to be added before the pro-adjective.

The required implications for the ambiguities presented for the sentence at the beginning of the chapter are presented below. For convenience, the "specification" and "action" tabulations are repeated.

The information regarding allowed intervening words in each instance is not stated in the "specification" column. This information must be furnished by a linguist by careful study of each individual problem.

Specification	Action	Equation
pro-adjective in genitive or locative; preceding preposition governing genitive case.	eliminate English prepositions	$\cdot_G \cdot p_n{}^{P} \cdot_G \longrightarrow \sim P$
present active participle; preceding agreeing adjective.	eliminate English prepositions	$A._{*}^{r.a.t}\ .pA.* \longrightarrow \sim P$
substantive; preceding agreeing adjective	eliminate English prepositions	$S.*\ .p^{A}.* \longrightarrow \sim P$
substantive in nominative or accusative plural or genitive singular; immediately preceding substantive suggests genitive singular.	eliminate (s) retain (of)	$S.\underset{(s)\cdot(of)}{\left[G_s\cdot(N.A)_{PL}\right]}\cdot\left\{A+V\right\}p_1S$
substantive in genitive with preceding substantive	retain (of)	$S._G\cdot\ \left\{A+V\right\}_p S \longrightarrow (of)$
multiple distribution class (particle, predicative adjective, preceding commas.) Particle is indicated.	eliminate "(is)final/finite(ly)" retain "of-course"	$[\text{конечно}]\ \cdot p_1K\cdot f_1K \longrightarrow \sim (is)\ final/finite(ly)$
verb, infinitive, with preceding finite verb which requires "to".	retain (to)	$V^{I}\cdot p(V\neg(to)) \longrightarrow (to)$
substantive in nominative-accusative plural or genitive singular; preceding substantive suggests genitive singular.	eliminate (s)	$S.\ \underset{(s)\ \cdot\ (of)}{\left[G_s\ \cdot\ (N.A)_{pl}\right]}\ \cdot\ p^{S} \longrightarrow \sim$

Once these expressions have been written for a particular linguistic specification, a flow chart may be constructed with relative ease from the expression. The computer algorithm then may be written in a straightforward manner.

This notation is certainly not unique. One Russian group[22] and the MIT[23] group have both developed systems of notation for expressing linguistic requirements in equational form.

The system of notation and logical expressions which are presented above have been found to be extremely efficient. Programming has become a straightforward process, and the chances of overlooking vital information are greatly reduced.

In the next chapter the actual programming of the IBM 650 will be described, and, as will be shown, the expressions presented in this chapter form an important part of the programming procedure.

[22] Panov, D. Yu., A. A. Lyapunov, and I. S. Mukhin; The Automatization of Translation from One Language to Another. Report distributed by Office of Technical Services, U. S. Department of Commerce, Washington 25, D. C., No. 59-11324, JPRS/DC-379.

[23] Yngve, Victor H., A Programming Language for Mechanical Translation, MECHANICAL TRANSLATION, Vol. 5, No. 1, pp. 25-41.

Chapter 5

Use of the IBM 650 for the Study of Syntax in the Solution of the Problem of Multiple Meaning

Linguistic analysis of text material can supply precise specifications for necessary cues to resolve multiple meaning and word-order problems of Machine Translation. It can be shown[24], however, that a problem solution may be completely specified; yet no algorithm can be constructed which will give the solution in a finite number of operations. Specification of the cues is not enough to assure that the particular problem can be solved by a computer; a program algorithm must be displayed.

In order to test the validity of the cues described in the last chapter and also to develop general programming techniques, program algorithms for these cues were written for the IBM 650 computer. The algorithms and their (processing) effectiveness are described in this chapter.

The routines herein described resolve some representative ambiguities of verbs, substantives, conjunctions, and one multiple (pro-adjective and pro-substantive) distribution class. In addition, the rule concerning the agreement of substantives with modifying adjectives and governing prepositions is exploited to solve certain problems of multiple meaning.

The IBM 650 presents certain storage problems when used for translation research. The 2,000-word storage capacity of the device is not only entirely inadequate for the translation lexicon but is also too limited to allow complete storage of all the processing programs described in this report. Consequently, a special procedure was devised. Instead of storing the translation lexicon in the computer memory and performing the dictionary search automatically, the lexicon is stored on a file of IBM cards. Dictionary search is then accomplished by hand. The dictionary search involved in translating a text passage proceeds by extracting from the card file the dictionary entry corresponding to each word in the text passage. It is necessary, of course, that copies of the cards be made for the individual dictionary entries since the same word will often appear several times in a particular text passage.

After this manual dictionary search is completed, the entries are stacked in text order, and the text passage is ready for processing.

Since it is not possible to store all programs which are described in this chapter on the memory drum at one time, the programs were divided into four parts. The first part, called the First Round of Processing, is concerned with the rules of agreement of the substantive with its modifying adjectives and governing prepositions. The second part, called the Second Round of Processing, is concerned with coordinating conjunctions which link substantives in various patterns. The third part, called the Third Round of Processing, is concerned with verb patterns and one multiple-form-class problem. The fourth part is called the Interpretive Routine. This program inserts English prepositions, whose function is expressed in the Russian language by inflectional endings, and performs certain other functions such as erasing unwanted equivalents in the individual entries.

To process text material, first the program and then the text card deck is loaded into the computer. The computer executes the programs and punches out another card deck which is just like the input text deck except for the processing accomplished by the program. The changes appear in modifications of the tags. The text deck from this round of processing is then placed immediately behind the program deck for the next round of processing, and both decks are fed into the machine again. This is repeated until all three rounds of processing have been completed. After the Third Round of Processing the output deck is fed into the computer with the Interpret Routine in the same way as before. After the Interpret Routine the output deck may be introduced into the accounting machine to print out the translation.

The First Round of Grammatical Processing

The First Round of Processing was originally entirely the work of the author. In its present form the First Round of Processing is also the work of Niehaus, who wrote the programs now used.

As stated previously, the First Round of Processing considers the rule that Russian substantives must agree in gender, number, and case with the adjectives which qualify them and that prepositions must agree in case with the substantives which they govern. As the name preposition indicates, these words always precede the substantive (and the adjectives and adverbs which modify the substantive) which they govern. It is also assumed that the adjective always precedes the substantives which it qualifies. Occasionally this does not hold true in Russian, but such instances will not be considered in the First Round of Processing since their solution involves more than a matching procedure. This assumption does not introduce any errors into the

[24] Davis, Martin; COMPUTABILITY AND UNSOLVABILITY, McGraw-Hill Book Co., Inc., 1958, p. xv.

solution; it only makes the solution incomplete.

The logical equations for the First Round of Processing may be written as follows:

$$(P+A+\forall)_{\cdot *}\cdot\{A\}_{\cdot *}\,f(A+\forall+S+\text{Ƨ})_{*}\longrightarrow\cap$$

$$(A+\forall+S+\text{Ƨ})_{\cdot *}\cdot\{A\}_{\cdot *}\,p(A+\forall+P)_{\cdot *}\longrightarrow\cap\cdot\sim P$$

where the notation is as described in Chapter 4.

The application of the equations of the First Round of Processing is best demonstrated by an example. In the sentence below a preposition is encountered; the three semantic units following it are, respectively, an adjective, another adjective, and a substantive.

из	новой	русской	школы
preposition	adjective	adjective	substantive
G	G_s, D_s, I_s, P_s	G_s, D_s, I_s, P_s	G_s, N_{pl}, A_{pl}

The notation is that of the case specifications defined in Chapter 4. For convenience the definitions are repeated below.

Where: G_s = genitive singular

D_s = dative singular

I_s = instrumental singular

P_s = prepositional singular

N_{pl} = nominative plural

A_{pl} = accusative plural

The computer considers only the tags, i.e., for the instance of "из" (out of) only the coded "G" would be considered, or for "новой" only the coded "G_s, D_s, I_s, P_s." For the three rounds of processing described in this chapter, the computer does not consider either the coded Russian or the coded English equivalents. The coded form of "G" and "G_s, D_s, I_s, $\overline{P_s}$" are, together with certain other information, denoted as "tags." Processing performed in the three rounds shows up as modifications of those tags. The Interpret Routine interrogates the final tag form and from this modifies the target equivalents.

According to the first equation, when a preposition is encountered, the machine is to go on to the following semantic unit. If the next semantic unit is an adjective or a substantive, the logical intersection is performed which delineates the area of agreement of both the preposition and the words governed by it. Since the preposition governs the genitive and the immediately following adjective has a proper genitive singular form, *the intersection gives the case for this particular example as genitive singular for each of the first two words.* Since in prepositional phrases adjectives are mostly nonterminal words, as denoted in the equation by the expression $\{A\}$ (see page 378), the machine then goes on to the next semantic unit. The next word also turns out to be an adjective; so the case intersection is performed again, giving also genitive singular for the case and number of the third word. Since the third word (an adjective) is also not necessarily a terminal word, the machine proceeds to the next semantic unit. The next word, the fourth, is a substantive. The case intersection is again performed and establishes that the case of this substantive is also genitive singular. Since substantives are not allowable intervening words, the machine stops searching.

The second equation is similar except that the search is directed to preceding and not to following words. In addition to performing case intersection, the second equation prescribes that, if an adjective, pro-adjective, substantive, or pro-substantive is preceded by an agreeing adjective, pro-adjective, or preposition, then no English preposition is to precede the target equivalent of the semantic unit being processed.

A complete print-out of the computer program for the First Round of Processing will be found in Niehaus' thesis.[25]

Following are three examples of text material actually processed by the computer:

1. a. Осуществление эталонов. Конструкция эталона, его физические свойства и способ осуществления определяются величины, единица которой воспроизводится, и состоянием измерительной техники в данной области измерений.

   b. The word-for-word translation of this material is as follows:
   Realization (of)standards. Construction/design (of)standard, (of)(to/for)(by/with/as) his/its//him/it physical-properties and/even/too method (of)realization (they)are-defined/determined/assigned (by/with/as)nature (of)magnitude/quantity(s), unit/one (of)(to/for) (by/with/as)which is-reproduced, and/even/too (by/with/as)state/fortune (of)(to/for) (by/with/as)measuring/dimensional (of)technics/practice//technologists in/at/to/on/of/like (of)(to/for)(by/with/as)given (of)(to/for)area/oblast(s) (of)measurements.

   c. The translation after the First Round of Processing is:
   Realization of standards. Construction/design of standard, (of)(to/for)(by/with/as)his/its/

[25] Niehaus, Udo K., Russian to English Translation Processing With an IBM 650 Computer, Master of Science Thesis, University of Washington.

him/it physical-properties and/even/too method (of)realization (they)are-defined/determined/assigned by/with/as nature (of)magnitude/quantity, unit/one (of)(to/for)(by/with/as) which is-reproduced and/even/too by/with/as state/fortune of measuring/dimensional technics/practice//technologists in/at/on (of)(to/for)(by/with/as)given (of)(to/for)area/oblast(s) of measurements.

The word-for-word translation contains 88 words while the processed translation contains 82 words. In addition, in six instances the program determined that English prepositions were definitely required; and, accordingly, the parentheses were removed from the prepositions.

The "information" contributed by the processing routines may be calculated with the defining equation (see Equation 8, Chapter 5).

$$I_i = \frac{\log_2\dfrac{(\text{number of possible sequences in the word-for-word translation})}{(\text{number of possible sequences in the processed translation})}}{\log_2(\text{number of possible sequences in the word-for-word translation})} = \frac{7.63}{43.15} = .177$$

2. a. Только определения независимых единиц дакт некоторую программу действий для осуществления эталонов, поскольку эти определения содержат указание на вещество и условия, в которых оно должно находиться.

   b. The word-for-word translation is:
   Only (of)definition/determination/attribute(s) (of)independent (of)units/ones give/allow certain/some program (of)act(ion)s/effects/operations for (of)realization (of)standards, in-as-much-as these (of)definition/determination/attribute(s) contain/maintain indication/instruction on/in/at/to/for/by/with substance and/even/too (of)condition(s) in/at/to/on/of/like (of)which it must/should/owes (to)be(found).

   c. The translation after the First Round of Processing is:
   Only (of)definition/determination/attribute(s) of independent units/ones give/allow certain/some program of act(ion)s/effects/operations for realization of standards, in-as-much-as these definition/determination/attributes contain/maintain indication/instruction on/in/at/to/for/by/with substance and/even/too (of)condition(s), in/at/to/on/of/like which it must/should/owes (to)be(found).

In this instance the word-for-word translation contains 62 words while the processed translation contains 58 words. The program also determined that in three instances the English preposition, "of", was actually required by the context; and in one case it established that the plural form of a substantive was required. The parentheses were accordingly removed in these four cases.

The amount of information contributed by the First Round of Processing may be calculated as follows:

$$I_i = \frac{8}{32.72} = .245$$

3. a. Последнее достигается умелым выбором частного случая общих законов явлений, связывакщих величины, единицы которых входят в определение производной единицы.

   b. The word-for-word translation is:
   Last/latter is-attained/reached (to/for)(by/with/as)skillful (by/with/as)choice/election (of)private/particular/quotient/partial (of)case/chance/occurrence (of)general/common/total (of)laws (of)appearances/phenomena/symptoms, (of)tying/connecting/knitting (of)magnitude/quantity(s), (of)unit/one(s) (of)which enter-into definition/determination/attribute (of)(to/for)(by/with/as)derivative (of)unit/one(s).

   c. The processed translation after the First Round is as follows:
   Last/latter is-attained/reached by/with/as skillful choice/election of private/particular/quotient/partial case/chance/occurrence of general/common/total laws of appearances/phenomena/symptoms, (of)tying/connecting/knitting (of)magnitude/quantity(s), (of)unit/one(s) (of)which enter-into definition/determination/attribute of derivative unit/one.

The third example illustrates a case where the First Round of Processing was very effective in improving the output. The word-for-word translation contains 62 words, while the processed translation contains only 40 words. The figure for the amount of information contributed by the First Round of Processing in this instance is:

$$I_i = \frac{12.81}{35.3} = .363$$

The Second Round of Grammatical Processing

The Second Round of Grammatical Processing is concerned with routines designed to resolve the multiple grammatical meaning problem of a large class of substantives and of certain coordinating conjunctions. The logical equations for the Second Round of Grammatical Processing are as follows:

1. $S_{\cdot *} \cdot p_1 C_{\cdot c} p_2 S_{\cdot *} \rightarrow \sim P \cdot \sim(S)$
2. $S_{\cdot *} \cdot p_1 C_{\cdot c} \cdot p_2 K \cdot p_3 S_{\cdot *} \rightarrow \sim P \cdot \sim(S)$
3. $S_{\cdot *} \cdot p_1 C_{\cdot c} \cdot p_2 K \cdot \sim p_3 S_{\cdot *} \{U\} p S_{\cdot *} \rightarrow \sim P \cdot \sim(S)$
4. $S_{\cdot *} \cdot p_1 K \cdot p_2 S_{\cdot *} \rightarrow \sim P \cdot \sim(S)$
5. $S_{\cdot *} \cdot p_1 K \cdot \sim p_2 S_{\cdot *} \cdot p S_{\cdot *} \rightarrow \sim P \cdot \sim(S)$
6. $S_{\cdot (G_s \cdot (N \cdot A)_{pl})} \cdot p_{1+2+3} N \rightarrow \sim P \cdot \sim G$
7. $S_{\cdot (G_s \cdot (N \cdot A)_{pl})} \cdot \{U\} \sim p S \cdot \{U\} f V_{pl} \rightarrow \sim G$
8. $S_{\cdot (G_s \cdot (N \cdot A)_{pl})} \cdot p_1 K \cdot f V_{pl} \rightarrow \sim G$
9. $S_{(G_s \cdot (N \cdot A)_{pl})} \cdot p_1 \forall_{N_{pl}} \rightarrow \sim G$
10. $S_{\cdot (G_s \cdot (N \cdot A)_{pl})} \cdot p_1 A_A \xrightarrow{r \cdot a \cdot t} \sim G$
11. $S_{\cdot (G_s \cdot (N \overset{\bullet}{\cdot} A)_{pl})} \cdot p_1 S \rightarrow \sim (N + A)$
12. $S_{\cdot (G_s \cdot (N \cdot A)_{pl})} \cdot \{A\} p B_1 \rightarrow G_s$
13. $S_{G_s} \cdot \{U\} p N \rightarrow \sim G \cdot (+S)$
14. $S \cdot \{U\} p N \rightarrow \sim P$
15. $S_1^{a} \cdot \{U\} p V_{r \cdot p} \rightarrow \sim(\text{with/as})$
16. $S_1^{c} \cdot p_1 G^{c} \rightarrow \sim(\text{by/as})$
17. $S_1 \cdot p_1 C_{\cdot c} \cdot p S_1 \rightarrow (\text{with/as})$
18. $S_1 \cdot p_1 S^{c} \rightarrow \sim(\text{by/as})$
19. $S_1 \cdot p_1 (S^{\cdot c}_{\cdot 1}) \rightarrow \sim I$
20. $S_1 \cdot \{U\} p (V \frown I) \rightarrow \sim I$
21. $S_{\cdot G} \cdot p_1 (D + \forall) \rightarrow \sim G$
22. $S_{\cdot G} \cdot \sim p_1 S \cdot \{U\} p V \rightarrow \sim G$
23. $S_{\cdot D} \{U\} p (V \frown \text{to}) \rightarrow \sim(\text{for})$
24. $S_{\cdot D} \{U\} p (V \frown \text{for}) \rightarrow \sim(\text{to})$
25. $S_{\cdot (G \cdot D)} \cdot \sim p_1 S_{\cdot (D + I)} \rightarrow \sim D$
26. $C_{c \cdot s} \cdot p_1 K \cdot (p_2 + f_2)_{\cdot *} \rightarrow C_c$
27. $C_{c \cdot s} \cdot p_1 K \cdot \sim (p_2 \cdot f_1)_{\cdot *} \cdot (p \cdot f_1)_{\cdot *} \rightarrow C_c$
28. $C_{c \cdot s} \cdot (p_1 \cdot f_1)_{\cdot *} \rightarrow C_c$
29. $C_{c \cdot s} \cdot \sim (p_1 \cdot f_1)_{\cdot *} \cdot p_2 K \cdot (p_3 \cdot f_1)_{\cdot *} \rightarrow C_c$

The development of the Second Round of Grammatical Processing is the result of the cooperation of Dr. Lew R. Micklesen, Mr. Udo K. Niehaus, and the author. Dr. Micklesen analyzed approximately twenty-five hundred running words of Russian text to determine the procedures necessary to solve multiple-meaning problems. These texts were then analyzed by the author for necessity and sufficiency. Logical equations were then written, and the flow diagram was drawn. A print-out of the program of the Second Round of Processing may be found in Niehaus' thesis.

The Second Round of Processing is inherently very different from the First Round. The First Round of Processing is essentially a matching procedure, whereas the Second Round involves search routines. In the First Round, when an adjective or preposition is encountered, the machine matches the grammatical tag with the tag of succeeding adjectives and the next substantive, performing the logical intersection of these tags. The succeeding rounds of processing involve search procedures; the context of the semantic unit is examined for specific word patterns which will indicate that one particular target alternative should be chosen in preference to others.

Since the second and succeeding rounds of processing involve search procedures for specific semantic units and their environment, it is necessary for the computer to store in the high-speed memory the text environment of the semantic unit being processed. If too few of the environmental semantic units are stored, the patterns cannot be established; and, consequently, the multiple-meaning problem cannot be resolved. If more than a sufficient number of semantic units and tags are stored, computer memory space is wasted. Thus it is necessary to determine roughly the extent of this "inter-word influence" in order to obtain an optimum amount of storage for semantic units.

To determine the extent of this "inter-word influence" a sample of about 150 sentences of text was analyzed in consideration of the logical equations. A typical example of the results is the following:

RANGE OF SEARCH FOR THE PROCESSING OF SUBSTANTIVES

sentence and word number	range of search
2-4	1 back
2-7	1 back
3-1	2 back, 3 fwd
3-5	1 back
3-9	2 back
4-1	1 back, 4 fwd
5-6	1 back
5-7	1 back, 2 fwd
6-5	7 back
6-9	1 back
6-14	1 back
6-15	1 back
6-25	7 back
6-42	1 back
6-48	1 back
7-1	1 back
7-13	1 back
8-4	1 back
8-10	1 back
8-11	1 back
9-5	2 back
10-1	1 back
11-12	1 back
11-16	1 back
11-17	2 back
11-18	2 back
13-3	1 back
13-4	2 back
17-6	2 back

For this specific example, if only two semantic units are stored at one time, the semantic unit being processed and the semantic unit immediately preceding, then 18 of 29 tests can be performed. If three semantic units are stored at one time, the semantic unit being processed and the two immediately preceding it, then six more of the procedures can be handled, or 24 of the 29. A storage of two more semantic units (forward) will increase the number of procedures which can be handled to 25. Increasing the forward storage by one more (six semantic units stored in all) will increase the procedures handled to 26. If the storage is increased to a total of seven semantic units, all procedures except those (6-5 and 6-25) which test seven semantic units back may be programmed. To consider these last two instances, the semantic unit storage would have to be increased by 5, or almost doubled.

When the program requires a search as far back as seven semantic units, the semantic unit being sought is nearly always a substantive or a verb; and the intervening material seems to consist, at least in part, of a parenthetical expression. Thus, in the above example, instead of increasing the running storage by 5 semantic units, a more efficient procedure would be to store the tag information concerning the last substantive and the last verb in some convenient memory locations. The following example will illustrate what is meant. Suppose that the machine is processing the sentence: От материала требуется прочность, стойкость иногда исключительная чистота.
A free translation of this sentence would be:

(The material must be durable, stable, and sometimes exceptionally pure.)

If the semantic unit being processed at a particular instant is "прочность" the storage locations in the computer will be filled as shown in Row I of the following tabulation:

STORAGE OF TEXT MATERIAL AT
DIFFERENT STAGES OF PROCESSING
(Target-Language Equivalents and Tags Are Not Shown)

PRECC	PRECB	PRECA	PRSNT	FOLLA	FOLLB	LVERB	LSUBS
от	материала	требуется	прочность	,	стойкость	.	.
требуется	прочность	,	стойкость	иногда	исключительная	.	материала

Where:

PRECC is the third semantic unit back in the sentence from the semantic unit being processed (preceding C).
PRECB is the second semantic unit back from that being processed (preceding B).
PRECA is the semantic unit immediately before that being processed (preceding A).
PRSNT is the word which is currently being processed (Present).
FOLLA is the succeeding semantic unit in the sentence, the next to be processed (following A).
FOLLB is the second semantic unit in the sentence after that being processed (following B).
LVERB is the last verb in the sentence which occurs before the semantic unit in PRECC (last verb).
LSUBS is the last substantive in the sentence which occurs before the semantic unit in PRECC (last substantive)

Note that commas (or any punctuation marks) are treated as semantic units.

When the processing for прочность has been completed, the computer enters the entry for comma in the PRSNT location. Since a comma requires no processing, the machine immediately moves стойкость into the

PRSNT location. In each of these movements, the other entries are moved one location backwards. When the processing of стойкость starts, the entries for от and материала have been erased, the entry for требуется has been moved from PRECA to PRECC, the entry for прочность to PRECB, and the new entries, those for иногда and исключительная have been introduced into FOLIA and FOLLB respectively.

Row 2 of the table shows the storage contents when the word is being processed. Note that the substantive, материала, is now stored in the LSUBS location.

An analysis of about 2500 semantic units of text indicated that the storage provided in this way (six consecutive semantic units and the last verb and the last substantive) allows solution of about 86% of the occurrences of problems which can be resolved by the processing of the Second and Third Rounds. About 1100 locations are still available for the processing routines after reserving the necessary storage for the text material, load and unload routines, and trace.

If enough computer storage had been allotted for the text to permit a complete processing of this sample by this method, the locations available for the processing routines would have been reduced by approximately 60%; or about 450 locations rather than 1100 would have been available. Since about 350 locations are used for the load and unload routines, then only 450 minus 350 or 100 locations would be available for the processing programs. Very little processing can be performed in 100 program steps.

Using the examples given earlier, the result of the two rounds of grammatical processing and the interpret routine is the following:

Example 1.

Realization of standards. Construction/design of standard, (of)(to/for)(by/with/as)his/its//-him/it physical-properties and/even/too method of realization (they)are-defined/determined/assigned by nature of magnitude/quantity, unit/one (of)(to/for)(by/with/as)which is-reproduced, and/even/too by state/fortune of measuring/dimensional technics/practice//-technologists in/at/on (of)(to/for)(by/with/as)given (of)(to/for)area/oblast(s) of measurements.

The amount of information extracted by the First and Second Rounds of Processing may then be calculated from Equation 1 thus:

$$I_i = \frac{\log_2 \frac{10^{13}}{3.41 \times 10^{8}}}{\log_2 \quad 10^{13}} = .344$$

and the amount of information in the Second Round of Processing is:

$$I = T_{output} - T_{input} = .344 - .177 = .167$$

Example 2.

Only definition/determination/attribute s of independent units/ones give/allow certain/-some program of act(ion)s/effects/operations for realization of standards, in-as-much-as these definition/determination/attribute s contain/maintain indication/instruction on/in/at/to/for/by/with substance and conditions, in/at/to/on/of/like which it must/should/owes (to)be(found).

The amount of information extracted by the First and Second Rounds of Processing may then be calculated from Equation 1 thus:

$$I = \frac{\log_2 \frac{7.09 \times 10^{9}}{11.6 \times 10^{5}}}{\log_2 \quad (7.09 \times 10^{9})} = .384$$

and the amount of information in the Second Round of Processing is:

$$I = .384 - .245 = .139$$

For example 3, the output after the completion of the two rounds and the Interpretation Routine is as follows:

Last/latter is-attained/reached by/with/as skillful choice/election of private/particular/-quotient/partial case/chance/occurrence of general/common/total laws of appearances/-phenomena/symptoms, (of)tying/connecting/knitting magnitude/quantity s, unit/one (of)which enter-into definition/determination/attribute of derivative unit/one.

The amount of information extracted by the First and Second Rounds of Processing equals:

$$I_i = \frac{\log_2 \frac{3.98 \times 10^{10}}{7.35 \times 10^{5}}}{\log_2 \quad (3.98 \times 10^{10})} = .466$$

and the amount of information in the Second Round of Processing alone is:

$$I = T_{output} - T_{input} = .466 - .363 = .083$$

The Third Round of Grammatical Processing

The Third Round of Grammatical Processing is concerned with representative problems of the intended meaning of verbs and one of the intended meanings of a multiple distributional class. The logical processing equations considered by the Third Round of Processing are as follows:

1. $M^{2\cdot\forall}_{\cdot k}\cdot\sim\{A\}_{\cdot k}\, f(A+S)_{\cdot k} \rightarrow \sim P\cdot\sim\forall$

2. $M^{2\cdot\forall}_{\cdot k}\cdot\{A\}_{\cdot k}\cdot f(A+S)_{\cdot k}\,\{A\}\,p\,P_{\cdot k}\rightarrow\sim P\cdot\sim 2$

3. $M^{2\cdot\forall}_{\cdot k}\{A\}_{\cdot k}\, f(A+S)_{\cdot k}\cdot\sim\{U\}\,p(V^{P}+V)\rightarrow \cap\cdot\sim S$

4. $V_{3pl}\cdot\{U\}\,p\,E^{N}\rightarrow\sim(\text{they})$

5. $V^{\sim T}_{3pl}\{U\}\,p\,E^{N}\cdot f\,E^{N}\rightarrow\sim(\text{they})$

6. $V_{3pl}\{U\}\,(\sim p+f)E^{N}\rightarrow(\text{they})$

7. $V^{I}\cdot\sim p\,V^{x}\cdot p(E^{P}+H^{P})[(A+S)\frown I]\rightarrow(\text{to})$

8. $V^{I}\cdot\sim p\,V^{x}\{E\quad H\}\,p(E^{I}\frown\text{to})\rightarrow(\text{to})$

9. $V^{I}\cdot p(V\frown\text{to})\rightarrow(\text{to})$

10. $V^{I}\cdot p(V\frown\sim\text{to})\rightarrow\sim(\text{to})$

11. $V^{G}\cdot(p+f)\,K\rightarrow\sim(\text{are})$

12. $V^{G}\cdot\sim(p+f)\,K\rightarrow(\text{are})$

13. $V^{f}_{1_{pl}}\cdot p\,S^{P}\cdot\rightarrow(\text{let-us})$

14. $V^{f}_{1_{pl}}\cdot\sim p\,S^{P}\cdot\rightarrow\sim(\text{we shall})$

15. $V_{3S}\cdot p\,S^{I}_{N}\rightarrow\sim(\text{is})\cdot\sim(\text{will})$

16. $V^{m}\cdot\sim p\,S_{N}\,f\,V_{I}\rightarrow(\text{it})$

The following examples will illustrate the application of the Third Round of Processing:

Example 1. After the First, Second, and Third Rounds of Processing

> Realization of Standards. Construction/design of standard, his/its physical properties and method of realization are-defined/determined/assigned by nature of magnitude/quantity, unit/one (of)-(to/for)(by/with/as) which is reproduced, and/even/too by state/fortune of measuring/dimensional technics/practice/technologists in/at/on given area/oblast of measurements.

The information in the First, Second, and Third Rounds for this example is thus:

$$I_i = \frac{\log_2 \dfrac{10^{13}}{7.25 \times 10^{4}}}{\log_2 \quad 10^{13}} = .626$$

and the information in the Third Round alone is:

$$I = .626 - .344 = .282$$

For Example 2.

> Only definition/determination/attributes of independent units/ones give/allow certain/some program of act(ion)s/effects/operations for realiztion of standards, in-as-much-as these definition/determination/attributes contain/maintain indication/instruction on/in/at/to/for/by/with substance and conditions, in/at/to which it must/should/owes be(found).

In this example the information content of the First, Second and Third Rounds of Processing may be calculated to be:

$$I_i = \frac{\log \dfrac{7.09 \times 10^{9}}{1.45 \times 10^{5}}}{\log 7.09 \times 10^{9}} = \frac{15.57}{32.7} = .475$$

and the information content of the Third Round will be:

$$I = .475 - .384 = .091$$

The First, Second and Third Rounds of Processing are all concerned with multiple-meaning problems which conform to the classical concept of "grammatical" problems. Obviously many multiple-meaning problems cannot be solved by tests of classical grammar. It is convenient to consider the information in routines which would solve all so-called grammatical intended meaning problems even though the distinctions "grammatical" and "non-grammatical" are somewhat arbitrary. In Example 1, the only remaining multiple-meaning problem which could be considered grammatical concerns the equivalent of которой "(of)(to/for)(by/with/as)which." There are seven possible sequences in this equivalent; so the information in a routine which would solve this problem would be

$$I_{GR} = \frac{\log 7}{42.88} \approx .065$$

For the case of Example 2, there are no further problems which can be solved by classical grammatical tests, thus

$$I_{GR} = 0$$

For Example 3, "--by/with/as skillful--" is not clearly a grammatical problem, so it will not be included. The two unresolved problems, "--(of)tying/-----" and "--(of)which--" conform to the intuitive idea of grammatical problems, so

$$I_{GR} = \frac{\log 4}{36.32} \approx .055$$

Table 1 illustrates for a larger sample the effectiveness of the three rounds of processing.

The table demonstrates that over 80% of grammatical problems are solved by the three rounds of processing. The remaining problems are essentially all of a type which conform to the concept of "nongrammaticalness."

Some of the remaining grammatical problems are relatively easy to program and would have been added to the first three rounds of processing if space had permitted. Others are more difficult and will probably only yield to approaches which must be used to solve "nongrammatical" problems. It does seem that at least 95% of the typically "grammatical" problems can be solved by routines of modest size.

Many problems could be treated more efficiently by increasing the computer text storage. This is especially true for problems concerned with the verb, since the subject and object of a verb are often considerably removed from the verb in the text. Thus, while "grammatical" problems are generally narrow context problems, the multiple-meaning problems of verbs frequently present a grammatical problem of wider context.

A print-out of the program of the Third Round of Processing is presented in Appendix 2.

The Interpret Routine

The interpret routine performs two functions: it inserts the English prepositions, whose function is expressed in Russian by inflections (see footnote, page 357), into the translation, and it executes the individual-entry subroutines, whose function must be explained. The reason for including the insertion of prepositions in the interpret routine is very simple: All rounds of processing narrow the number of case possibilities, hence the insertion of the prepositions must be postponed until final processing.

Individual-entry subroutines are processing programs which apply to one entry alone, and are consequently stored as an integral part of the individual entries. The most common example is that of deletions. For example, the Russian preposition "в" may either govern the locative or the accusative cases. The English equivalent of "в" is in/at/on/to/of/like. If, in a particular instance, "в" governs the locative, the alternatives to/of/like may be deleted. Since this deletion applies only to the entry for "в" it would be wasteful of general programming storage to include the deletion routine for "в" in the general program. In the procedure described in this thesis, the deletion program for "в" is stored with the entry in the large lexicon and is executed during the interpret routine. If, during the three rounds of processing, the case governed by "в" has been narrowed down to locative, the deletion routine eliminates to/of/like from the translation.

The reason that the individual entry subroutines are included in the interpret routine is that the entries are stored randomly during the Second and Third Rounds of Processing, and since the individual entry subroutines are stored as an integral part of the entry, the subroutine is also stored randomly during these two rounds of processing. In the interpret routine the entry is stored in a fixed location. Randomly stored routines are difficult to execute with the one-plus-one address system used on the IBM 650. The complication arises from the fact that the one-plus-one address system includes the address of the next program step to be executed (the instruction address) in each program step. If a routine is stored randomly, each instruction address must then be modified every time the entry is stored since in general the routine will be stored in a different place each time.

If the IBM 650 used a single-address system, the problem of random storage of routines would become much easier to solve since then the machine would automatically go to the next storage location for its next instruction, except for BRANCH and TRANSFER operations.

The individual-entry subroutines could have been transferred to a standard location before execution, but this would have created two other problems: first, storage would have to be reserved for this standard location, and second a transfer routine for initiating this transfer would have to be included. The solution of both of these problems would require unnecessary storage space; hence all individual entry subroutines were executed in the interpret routine.

No context is stored during execution of the interpret routine; processing is accomplished on a word-by-word basis. Since no syntactic interrogations are made during execution of interpret routine, no credit for improvement of the translation is awarded the interpret routine.

Three instances were encountered in the analysis of a Russian text in which it would have been convenient to perform the complete processing of a particular text word by an individual-entry subroutine. One example is the following:

[есть] $\{U\} \quad (f_n + p_n)(S_N \cdot \sim V^A) \longrightarrow \sim$ (to)eat

This implication could not be executed in the interpret routine since no context is stored. Since the implication applies to alone, a routine for accomplishing this processing is not general enough for inclusion in the general programs. Routines of this type were therefore not written for the study described in this report.

As was stated near the beginning of this chapter, the three rounds of processing operate only with the tags, and the results of the processing are manifested by modifications of these tags. The interpret routine then examines the modified tag and performs modifications to the target equivalents. In some instances processing requires examination of the source-language word. For example, one of the implications for ЧТО is as follows:

$$M^{C_s} \cdot S \cdot S^{I} \quad \cdot \quad p\ [\text{столь}] \longrightarrow C_s$$

Where: all notation is as defined in Chapter 4, with the addition of -- interrogative prosubstantive.

This implication requires searching for the occurrence of

Implications of this type were not included in the programs described; all processing routines of this report operated only with tags. This implication could have been included in either the Second or Third Rounds of Processing without any modification to the master programs. The only reason for not including implications in the general programs was the fact that other programs seemed more general.

A print-out of the Interpret Routine may be found in Niehaus' thesis.[25]

The Nongrammatical Intended Meaning Problem

Machine translation research requires an approach to the study of language which differs in some respects from classical linguistic analysis. It should reasonably be expected, therefore, that some specifications which result from machine translation research may add new concepts to the grammars of languages. One example is that of the distinction between "grammatical" and "nongrammatical" meaning which we have found convenient for our purposes.

Hill and Niehaus[26] have investigated one property of language which must be considered as "nongrammatical" at the present. The following discussion of the Hill-Niehaus method is included here for three reasons: first, the University's IBM 650 computer was the fundamental research tool in this work, second, the method is a very promising approach to some of the problems of intended meaning, and third, the extensions of the method which are proposed in the following paragraphs constitute original research by the author.

The Niehaus-Hill method is only concerned with substantives. The method requires that the entire field of science and technology be divided into fields and sub-fields. Each main field is assigned a two-digit number which always ends in zero, while each sub-field is assigned a two-digit number in which the left-hand digit is identical to the main field classifications, and the right digit denotes the particular sub-field. As an example, the digit 20 would apply to the entire field of physics, while 23 would apply to the sub-field of electricity and magnetism. Figure 2 displays the Hill Synoptic Table.

Niehaus mechanized this technique of field classifications in the Second Round of Grammatical Processing and the Interpret Routine. This processing is not included in the present version of these routines. The Niehaus procedure was as follows: During the Second Round, the field classification tags stored with the individual alternatives were examined, and a tabulation analogous to Figure 2 built up in the computer. For instance, the alternative "transistor" had a tag 83, indicating that this is a specialized word in the field of electrical engineering. Thus, when the tag associated with "transistor" was encountered, a "1" was added to the chart in position 83 and also in position 80. After the entire sentence was scanned, all tags were re-examined in the Interpret Routine. Alternatives with tags indicating fields not compatible with the table which was built up on the first scan were eliminated.

According to Niehaus this very simple routine was able to eliminate about 8% of the superfluous alternatives with excellent reliability.

The Niehaus-Hill method has two serious limitations. First, the synoptic table is two-dimensional with one dimension as the main field classification and the other as the sub-field classification. Second, the chart is divided into "field areas," which are defined historically rather than logically. For instance, "electricity and magnetism" is included in the general field of physics. No one would argue that this is one important area where the physicist must be well-grounded, but modern practice includes in the field of electrical engineering all problems concerned with circuit-theory research as well as all problems requiring application of Maxwell's equations. Electricity and magnetism and electrical engineering, however, are considerably separated in the synoptic chart.

In order to place accurately the various alternatives, one must first establish clearly defined reference points. Two reference axes (as in the Niehaus-Hill Synoptic Table) are clearly not sufficient. Also "fields of science" are not distinctive enough for accurate reference.

Since words are the entities to be specified, it would seem that specific words should be used as the references. The words which are chosen as the references should be as highly specialized words as possible so that they will exist as "points" in "semantic space." It would be ideal if these reference words could be few in number and so separated from each other semantically that the probability of any two of them occurring in the same text would be zero. The tremendous number of words which exist in a language, and consequently the very low probability of occurrence of any one word, makes the "single-word reference" impractical. A more realistic reference would be a group of highly specialized words in a particular area. For instance, "klystron," "magnetron," "traveling wave tube," "microwave," might be considered as one reference. Similarly, "cranium," "trachea," "upper lachrymal foramen," might be considered as another reference.

[26] Niehaus, Udo K., Automatic Pinpointing of Intended Non-Grammatical Meaning, op. cit. in footnote 1.

	1	2	3	4	5	6	7	8	9
0	Math.	Physics	Chemistry	Biology	Medicine	Social Sciences	Integrated Sciences	Applied Sciences	Technology
1	Algebra	Classical and Fluid Mechanics	Physical	Botany	Structure	Anthropology	Astronomy	Mechanical	Machinery Mechanism Tools
2	Geometry	Statistical Mechanics and Thermo	Inorganic	Zoology	Function	Linguistics	Geophysics	Thermo & Heat Engines	Production and Mfg. Methods
3	Analysis	Electricity and Magnetism	Analytical	Microbiology	Diagnosis	Philosophy	Geology	Electrical	Transportation
4	Statistics	Optics Spectra	Organic	Biophysics	Therapy	Sociology	Geography	Aeronautical Acoustic	Structures Architecture
5	Numerical Analysis	Quantum Mechanics	Biochemistry	Psychology	Pharmacy	Pol. Sci. Diplomacy	Meteorology	Nuclear	Mining Metals Ceramics
6	Relativity	Solid State	Photochem.	Agriculture and Forestry	Public Health Sanitation	Social Planning	Oceanography	Control	Marine and Naval
7		Nucleonics	Electro-	Animal Husbandry	Psychiatry	Economics Theory & Applied		Optical-Photo	Military Science Tactics
8		Metrology	Chem.Eng.	Fisheries	Veterinary	Legal		Materials	Textiles Paper
9									

Note: A single subscript in combination with a following zero denotes a whole field, i.e. 20 for physics. Wherever there is a subfield, the zero is replaced by the corresponding unit digit, i.e., 23 for electricity and magnetism.

FIGURE 2. SYNOPTIC CHART FOR SCIENCE AND TECHNOLOGY

After a suitable number of these references have been selected, an empirical try must be determined for each alternative to give the position of the alternative in this "semantic space" with respect to the "reference words." As text material is being processed, the tags of the alternatives would be tested, and the most probable position of the "semantic manifold" of the text determined. Alternatives whose tags indicate locations too far removed from the position of this "semantic manifold" would then be erased.

Reference dimensions might also be added for such things as concreteness or abstractness, animateness or inanimateness, and dimension.

As an example, consider the following translation:

skull/body shortened in/on/of frontal-orbital area/oblast, so that nasal-bones attain/-reach or almost attain/reach own back/posterior/rear ends/ways/leads of line of front/main edges/regions of orbits. (Figure 23)

In this sentence the co-occurrence of "frontal-orbital" and "nasal-bones" greatly increases the probability that "skull" and not "body" is required. Also, no reference to politics or government but reference to terms like "frontal-orbital" and "nasal-bones" would greatly increase the probability of "posterior" as against "back" or "rear" and "area" rather than "oblast." Because of implied dimensions the co-occurrences of "front/main" and "edges/regions" would indicate either "front edges" or "main regions" as most likely, while the occurrence of "line" would further increase the probability of "front edges" as well as "ends" over "ways" or "leads," again because of dimensional compatibility. The translation would then be:

"skull shortened in/on/of frontal-orbital areas so that nasal/bones attain/reach or almost attain/-reach own posterior ends of line of front edges of orbits. (Figure 23)"

Note that all these problems except the dimensional could have been solved by the Niehaus-Hill method. This is true because of the large number of highly specialized words in this sentence. Usually this amount of information is not available, and the solution will require the use of the probabilistic aspects of words. An individual word will not entirely establish but only imply the "semantic location." This is especially true for "narrow context" "nongrammatical" problems such as dimension compatibility.

It is the opinion of the author that a careful probabilistic study of the "nongrammatical" problem along the direction described is likely to be a profitable step in processing. The main difficulty currently faced by the University Research Group is that the IBM 650 is completely inadequate for this task.

Summary and Conclusions

Table II shows ten typical sentences in the form they would have if all multiple-meaning problems had been correctly solved but with no word-order rearrangement. The translation is unconventional, but readable. An obvious corollary is that the word-order problem is secondary to that of multiple meaning.

The processing performed on the IBM 650 computer was concerned with problems which conform to the classical concept of "grammatical" problems. As the distinction "grammatical" or "nongrammatical" is artificial, it is helpful to consider other classifications. One is that of wide-context versus narrow-context problems. The IBM 650 routines consider narrow-context rules of syntax. Another classification would be that of problems pertaining to entries versus problems which pertain only to individual alternatives. The routines consider those which could be solved by considerations of the tag describing the syntactic characteristics of the entry.

The routines are quite effective in resolving the "grammatical, narrow-context, entry-characteristic, multiple-meaning problems." From Table I it may be seen that over 90% of these problems were solved in the sample texts.

Certain other tests are easily postulated for resolving some of the remaining "grammatical" ambiguities, but other of these "grammatical" problems become extremely involved. A 95% solution of the grammatical problem would seem attainable by routines of the type described in this chapter, provided that entire sentences could be processed at one time by the computer.

Further work should utilize a larger computer, such as an IBM 704. The processing routines should be rewritten for this larger machine, and new routines added. The Niehaus-Hill method may be capable of sufficient extension to allow the solution of many of the problems that are not grammatical at the present state of linguistic knowledge.

TABLE I

Translation Quality of Sentences After Third Round

Sentence	Tq	T'q	Tq/T'q
1-2	.65	.07	.93
1-3	.56	.56	1.00
1-4	.40	.51	.78
1-5	.51	.56	.91
1-6	.42	.48	.89
1-7	.04	.05	.08
1-9	.63	.77	.82
Av.	.54	.58	.93

Tq = Translation Quality after third round

T'q = Translation Quality if all "grammatical" problems are solved.

TABLE II

1. Design of standard, its physical properties and method of realization are determined by nature of quantity, units of which are reproduced, and by state of measuring technics in given are of measurement.

2. Only definitions of independent units give some program of action for realization of standards, in as much as these definitions contain indication of substance and condition, in which it must be found.

3. Definitions then of derivative units pursue singular aim to establish sizes of units, not touching methods of their practical realization.

4. With cabbage butterfly after short interval of time from moment of ending of feeding of caterpillars before pupation losses of weight on average attained 25.9%

5. With lesser apple worm maximum weight of caterpillar had before ending of feeding, when average weight for one specimen attained .09320 grams.

6. A somewhat different picture of changes of weight was observed with soldier bug and with sorrel leaf eater.

7. With soldier bug reduction of weight came without sharp jumps which apparently is connected with feeding of bed bugs on warm days before autumn, then with leaf cutting beetle after cessation of feeding before disposition of beetles on wintering weight of their bodies fell for 5 days by 10.5% (from .01434 to .01282 grams by beetle).

8. History of study of tectonics of Alps played in geology is completely exclusive role.

9. In 1875 year appeared book of Suess "Origin of Alps."

10. It is possible to consider, that overrunning of folds depends on local tectonic causes, and namely from height of position of one or another of massive, serving as obstacle for spreading of folds.

Chapter 6

Design Specifications for a Digital Computer for Language Translation

In this chapter the translation problem will be analyzed in order to determine some of the design specifications of a computer for language translation. In order to specify completely the design of such a computer, the following information must be supplied:

1. The average amount of information which must be stored in each entry of the translation lexicon.
2. The number of entries which must be included in the translation lexicon.
3. The maximum amount of text which must be available to the processing programs at one time.
4. The required amount of program storage.
5. The required minimum processing speed.

These five requirements will now be considered:

Size of the Entry Storage

The amount of storage required for each individual entry varies considerably from one entry to another. The Russian part of the entry may vary from a single letter to considerably more than twenty letters. The English equivalents may vary from a few letters to more than forty-five. The tag part of the entry, where the grammatical and nongrammatical information is stored, has a standard length of 100 bits at the present stage of development. Little effort has been expended in optimizing the form or coding of the tags; so, although the amount of information included in the tag will need to be increased above the present level as processing becomes more sophisticated, the tag length will not necessarily have to be increased proportionately with the information stored in the tag. As a matter of fact, recoding the present 100-bit tag in a wholly utilized code, would reduce the length from 100 to less than 15 bits. The current tag form is used because it is more convenient for programming than a tag form using a wholly utilized code.

Average values of the different components of the individual entry may be estimated as follows:

1. Source-language storage--one word averaging seven letters. Since it is necessary to code both upper-case and lower-case letters, numerals, and about 15 different punctuation marks, a seven-bit (128 code) system is required. An average of seven characters per word would make a total of forty-nine bits.

2. English alternatives--average about two and one-half words per entry. The average English word is about five letters; so with allowance for word separation about fifteen letters, or about one hundred bits, will be required.

3. Tag storage can only be roughly estimated since the individual entry requirements have not been sufficiently explored. At present a single, 100-bit tag is assigned to each entry. This 100-bit tag would easily be reduced in size to 15 bits or less, as was pointed out previously, by using a wholly utilized tag code.

Considerably more information must be stored in the tags to meet the needs of the processing routines as routines become larger and more sophisticated. These additions should be largely overcome by more efficient coding; so the tag storage per entry is conservatively estimated to be as follows:

minimum--100 bits
maximum--200 bits

4. Individual-entry subroutines are convenient for processing applications where the routine applies uniquely to one entry and no other. The use of these individual-entry routines was discussed in detail in Chapter 5. The individual-entry subroutines may become large and may be used frequently. For deletions (see page 388), the experience of Niehaus and the author indicates that 8 to 14 program steps are required for each entry, with an average of about 10 steps. A count of several hundred entries disclosed that nearly half have a deletion problem. Hence the average length of deletion subroutines would be: $\frac{10}{2} = 5$ program steps per entry.

As an example of a more complex individual-entry subroutine, the implication for есть is as follows:

$$(\text{есть}) \qquad (p_n + f_n)\, s_n \longrightarrow \sim((\text{to})\text{eat}) \qquad (1)$$

Here the number of required program steps increases to 25 or 30.

In the analysis of the sample of 2500 semantic units, 46 implications were written; three of these pertained to particular semantic units, one to есть another to следует and the third to оказывается These words occurred only four times in the sample. Since there were about 1000 different semantic units in the sample, and since about 30 program steps are required to process implications such as (1), the individual-entry processing will require on an average of: $30\ \frac{4}{1000} = .12 \simeq 0$ program steps per entry. The total will then be approximately the requirement for processing of deletions, or five program steps per entry.

In Chapter 9 it will be demonstrated that the instruction[27] word length should be about 21 bits; so the storage requirements are approximately: 5 (21) = 105 bits.

It is possible that the use of subroutines will increase beyond this figure. Since it is possible to optimize the deletion routines by a more sophisticated technique than is used at present, a conservative estimate of the individual entry subroutine storage is not less than 250 bits and nor more than 350 bits.

Summarizing:

Total Text Storage Requirements

Average Per Entry	50 bits
Source language storage	50 bits
English alternatives	100 bits
Tag storage	100-200 bits
Individual entry subroutines	250-350 bits
Total	500-700 bits

Required Lexicon Storage

The lexicon is the translation dictionary. If dissection of compounds is not provided, each unique source-language semantic unit must have a separate entry; and each entry must contain all unique target-language equivalents of that source-language semantic unit together with all grammatical and nongrammatical information required by the processing programs. The entry must also contain any individual-entry subroutines which are required.

The translation lexicon contains a large amount of data arranged in much the same manner as an ordinary dictionary. Just as alphabetization and thumb indexing is used to minimize the search time for entries in an ordinary dictionary, the translation lexicon must be constructed in an optimum manner in order to make the machine translation as efficient as possible.

The most critical single problem is that of the required size of the translation lexicon. In this section the required size of this component of the translator is considered. The University of Washington MT operational lexicon is used as the basis for the following discussion.

The University of Washington translation lexicon consists of approximately eight thousand paradigmatic families, which were expanded to the present level of about 170,000 individual paradigmatic forms. Thus the ratio of paradigmatic families to paradigmatic forms is about 20 to 1.

The translation project has analyzed TM 30-545, "Russian-English Electronic Dictionary" to determine how much the present lexicon must be expanded in order to include all specialized semantic units in the field of electronics. This manual consists of approximately 22,000 technical terms extracted from an extensive corpus of Russian technical literature and is believed to be comprehensive. The count is complicated by the fact that more than half of the entries in TM 30-545 are not single-word entries but idiomatic sequences. Many of these sequences may be translated readily on a single word basis; others will best be handled as idioms. The count shows that about 7,000 sequences will have to be added, together with about 2,000 new individual semantic units. Micklesen estimates[28] that about 10 paradigmatic forms will be required for each of these 9,000 semantic units. Hence about 90,000 additional entries will be required in order to translate completely in the field of electronics. The total will then become: 170,000 + 90,000 = 260,000 entries.

The expansion required to include the specialized semantic units in other fields will depend greatly on the similarity of that field to electronics. If the other field were physics, the expansion would probably be limited to thirty thousand entries; while if it were organic chemistry or biology, the expansion would probably require another eighty thousand entries, plus a good compound dissection scheme for organic chemistry.

If all specialized words of electronics are included, the total requirement for the translation dictionary would be in the order of:

260,000 (500) = 130,000,000 bits minimum
260,000 (700) = 182,000,000 bits maximum

If two 100,000,000 bit Photoscopic Memories could be incorporated in a single machine, an adequate storage should be available for the scientific general-language word list plus the specialized word lists of electronics, and probably those of physics and mathematics as well.

We may speculate as to the ultimate size of a translation lexicon with reference to the size of one of the larger Russian dictionaries. Ushakov[29] lists about 80,000 entries, most, but not all, of which represent paradigmatic families. Naturally, a considerable number of Russian words exist which do not appear in Ushakov, and hence the 80,000 entries of Ushakov undoubtedly could be expanded to about 160,000 entries if a careful search of Russian technical and scientific literature were made. If the 160,000 entries are expanded to all their paradigmatic forms, with a liberal 20 to 1 ratio assumed, an estimate of the ultimate dictionary size would seem to be approximately:

[27] An instruction word is a coded sequence by means of which the programmer instructs the computer as to what operation the machine is to perform, what to perform this operation on, and sometimes where the computer is to obtain the next instruction.

[28] Micklesen, Lew R., op. cit., p. 74.

[29] Ushakov, D. N., EXPLANATORY DICTIONARY OF THE RUSSIAN LANGUAGE (In Russian). Reproduced for the American Council of Learned Societies, by Edwards Brothers, Inc., Ann Arbor, Michigan.

$$20\ (160{,}000)500 = 1{,}600{,}000{,}000 \text{ bits minimum}$$
$$20\ (160{,}000)700 = 2{,}140{,}000{,}000 \text{ bits maximum}$$

Required Amount of Text Storage

The "text storage" refers to the high-speed memory of the translation machine, where the entry for the word which is being processed is stored together with sufficient context to allow syntactic analysis. In this section an estimate will be made of the required size of the text storage.

Since the translation is to proceed on a sentence-by-sentence basis, the text storage must be at least large enough to store the entries for the longest sentence which will be encountered. Adequate additional storage must be provided so that, after the processing is completed for one sentence, the next sentence is immediately available for processing.

The longest sentence which has been encountered by the main project consists of 94 Russian words. If punctuation marks are treated as words, as they are in the procedure described in this report, the total entry count for this sentence is 113 words.

To determine the distribution of sentence length, a word count of about 150 Russian sentences was made. In this count, punctuation marks were treated as words. The results of this word count are displayed in Figure 3 with sentence length plotted against the number of occurrences of sentences of that length.

The plot of Figure 3 shows a contour very similar to the response of a second order, critically damped system. The equation for such a condition is:

$$f(x) = kx\,e^{-ax} \qquad (2)$$

where: k, a are constants

There are several ways to evaluate the constants in order to match the equation against the experimental data. The following calculations evaluate the constants by specifying that the median sentence length, x, shall be 22 and that the total area shall be 1. The latter condition is used for convenience and also because of the fact that the equation derived will later be used as a probability density function for estimating the optimum text storage size. Using Equation 2 and the conditions of the last paragraph:

$$\int_0^\infty kxe^{-ax}\,dx = 1$$

$$\left[\frac{ke^{-ax}}{a^2}(-ax-1)\right]_0^\infty = \frac{k}{a^2} = 1$$

Hence: $k = a^2$

$$(2)\ f = \int_0^{22} a^2\,x\,e^{-ax}\,dx = .5$$

$$\left[\frac{a^2e^{-ax}}{a^2}(-ax-1)\right]_0^{22} = \frac{-22a-1}{e^{22a}} = .5$$

Hence: $a = .0764,\ a^2 = .00583$

Equation 2 may then be written:

$$f(x) = .00583\ x\ e^{-.0764x} \qquad (3)$$

This equation is plotted on Figure 10 along with the experimental curve and is seen to give a good fit. The amplitude of the curve was adjusted to give the same area under the curve as the plot for Russian sentence length.

Other expressions exist which will also approximate this experimental curve. For instance, the function

$$y(x) = \frac{kx}{x^n + b} \qquad (4)$$

will give a good fit for proper choice of the constants. After some experimentation, values of $n = 3.9$ and $b = 1.43 \times 10^5$ were chosen for another plot of Figure 3. In the plot, the amplitude was also adjusted to give the same area under the curve as that for the plot of the length of Russian sentences.

There is a close correlation between the curve for the length of Russian sentences and that for the hyperbolic function,

$$y(x) = \frac{kx}{x^{3.9} + 1.43 \times 10^5}$$

This is another example which bears out Zipf's law; for if the log of both sides of the equation for y(x) is taken; then

$$\log(y(x)) = \log kx - \log(x^{3.9} + 1.43 \times 10^5)$$

If x is large enough this becomes:

$$\log(y(x)) \approx -\log x^{2.9} = -2.9 \log x$$

If the logarithm of y(x) is plotted against the logarithm of x, a straight-line plot results in accordance with Zipf's law.

The function y(x) does fit the experimental data better than the function f(x). In the following calculations, however, the function f(x) is chosen since f(x) shows a larger occurrence for high values of x than does y(x). *This will give conservative results in the calculations which are presented in the next few pages.*

The generating function expressed by Equation 3 relates sentence length, x, to the probability density function, f(x), of the occurrence of a sentence of that length. The problem which is to be solved in this section is the following: to find the number of consecutive text words which must be stored so that at any instant the probability is essentially 1 that two complete sentences are contained in the store. It will be assumed that this problem is equivalent to the following:

Given some ϵ, $1 > \epsilon > 0$, select an n such that for two randomly selected sentences, the first of which contains "a" words and the second "b" words,

$$p\ ((a + b) < n) \geq 1 - \epsilon.$$

The model does not strictly satisfy the conditions of the original problem. Sentence lengths are not entirely random, either for the same author or for different authors. An author who writes an extremely long sentence probably has a penchant for such structures. On the other hand, good form does demand a variety of sentence lengths. Since the generating function which was chosen will give a greater probability for long sentences than the actual empirical data indicates, the calculations which follow will be conservative. In other words, *if a temporary store is designed according to these calculations, the actual operation will be somewhat better than is indicated.*

First, the joint probability will be calculated for two randomly selected sentences having combined lengths less than or equal to some specified amount, L. From equation 3, this probability may be shown to be:

$$P = a^4 \int_o^L xe^{ax} \int_o^{L-x} ye^{ay}\, dydx$$

$$= a^4 \int_o^L xe^{ax} \left[e^{al} e^{-ax} (aL-ax-1) + \frac{1}{a^2} \right] dy$$

$$= a^2 e^{aL} \left[\frac{aL^3}{6} - \frac{L^2}{2} \ \frac{L}{a} \ \frac{-1}{a^2} \right] + 1$$

Where, from Equation 3:

$$a \approx -.0764$$
$$L \approx \text{some specified amount}$$

Since one sentence consisted of 115 words and punctuation marks, it is interesting to calculate the probability of two randomly selected sentences having a combined length of 115 entries.

$$P = 1 - \frac{1}{11.46} \ \frac{(.0764)\ \ (150)^3}{6} + \frac{(150)\ \ (.0764)^2}{2} + 150\ (.0764) + 1 = .9964$$

The probability that two randomly selected sentences having a combined length of more than 150 entries is then about four-tenths of one per cent, and that they will have 115 entries is about 2.7%. It would seem that storage for 115 entries is about the allowable minimum, and certainly storage for 150 entries should be adequate.

The required temporary text storage is therefore specified as:

minimum 500 (115) = 57,500 bits
maximum 700 (150) = 105,000 bits

Required Program Storage

The processing programs are required to improve the quality of the translation over that obtained by word-for-word translation. The size of the totality of processing programs is thus determined by the amount of improvement desired and also by the ingenuity of the linguist and the programmer. In this section an estimate of the size of the processing programs will be made.

The programs which were written by the author and Niehaus for the investigation described in this report totaled about 2,750 program steps. The IBM 650 requires ten digits per instruction, and the bi-quinary coding requires seven bits per digit. *The total storage requirement is thus* (7)(10)(2750) = 193,000 bits.

A machine designed for logical programming would not require 2750 program steps to perform the processing which was accomplished on the IBM 650. The IBM 650 routines required 350 steps for the load and unload routines. For the tentative design of this report, however, no storage is required since these operations are to be performed by logical circuitry. The main shortcomings of the IBM 650, however, occur first because the bi-quinary code gives an inconvenient computer word, and second because the important operation of logical AND and OR are not included in the IBM 650 operation codes. In Chapter 9 it will be shown that a special-purpose machine with binary operation and the AND and OR operation codes will allow a considerable reduction in the number of program steps below that required for IBM 650 operation. It is likely that such a special-purpose machine would perform the same processing of these 2750 program steps in about 1100 to 1200 steps.

The ultimate storage requirements for the processing routines can only be estimated. Obviously, the simple routines written for the investigations of this report are only a small part of the total programs which will be required for 100% translation. Yngve[30] estimates that ultimately the routine storage will be as extensive as the lexicon storage. The truth of this statement depends on what is defined as routine storage. In a careful analysis it is often quite arbitrary as to what is to be considered as dictionary material and what is to be considered as components of the processing routines. If the individual-entry routines and addresses of locations where deletions are to be made under specified conditions are considered as parts of the processing information, equal orders of magnitude are likely to be approached. As the programs get larger, however, sophisticated programming techniques give such a wealth and diversity of subprograms that adding another routine, even though quite complex in theory, usually results in only a short routine for initializing[31] a programming path through various subroutines which are already included in the master program.

Required Speed of Processing

The processing speed of the translation should be fast enough to justify economically the operation. In Appendix 4 it is shown that for an IBM 704 size computer, which would be about the same cost as the translation computer, the machine would have to process material at a rate of about 20 words per second in order to produce the gross income necessary for a self-supporting installation. The specification of the minimum processing speed which is required is then established as 20 words per second.

The design specifications of the translating computer are summarized in the next section.

Design Summary

The design requirements, or design specification, are summarized in Table 1:

TABLE 1

Design Requirements

	Minimum	Maximum	
Temporary Text Storage (high speed)	59,800	105,000	bits
Dictionary Storage (low speed)	114,000,000	2,140,000,000	bits
Program Storage (low speed)	27,500	60,000	Program steps
Minimum Processing Speed	20 words per second		

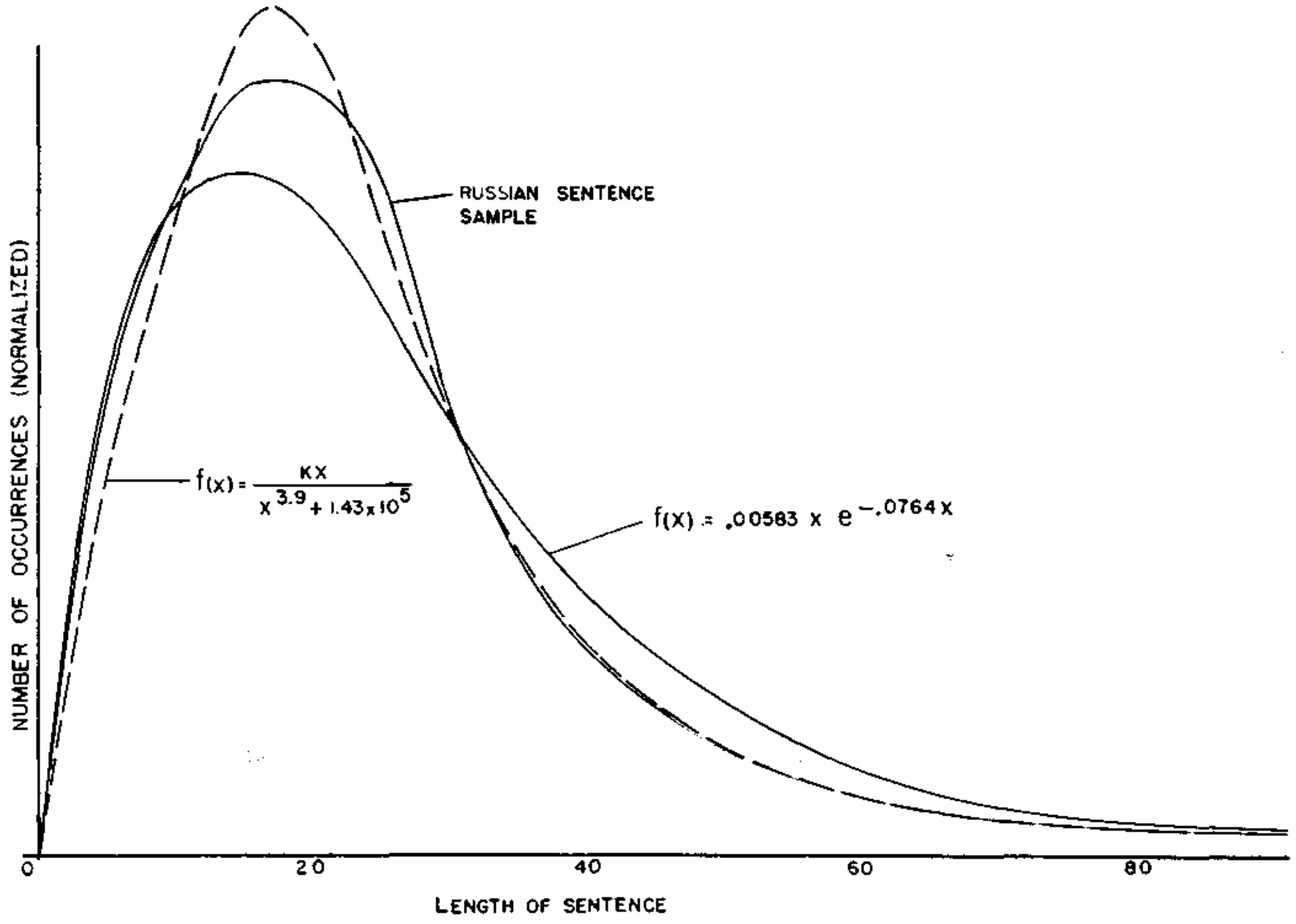
Table 1 specifies the design requirements for the tentative translator which is presented in the next chapters. The design exhibits considerable flexibility since the requirements stated are only estimates of values which cannot be specifically defined at this time.

[30] Yngve, Victor H., Syntax and the Problem of Multiple Meaning, MACHINE TRANSLATION OF LANGUAGES, edited by Locke and Booth, John Wiley and Sons, New York, 1955.

[31] "Initializing" is probably best explained by an example. Suppose subprograms "A" and "B" are stored randomly, and that the computer is to execute subprogram "A" and then subprogram "B". In order to get into "B", the computer must note the location of the entry step in"B" and store this location number in the exit step of "A". This operation is called "initialization".

LENGTH DISTRIBUTION OF RUSSIAN SENTENCES

FIGURE 3



Chapter 7

Design of a Large Memory for a Translation Computer

In Chapter 6 it was pointed out that a memory with a capacity of 130,000,000 to 182,000,000 bits and having a random access time of about 1/20 seconds is required to store the translation lexicon. In this chapter a tentative design of a memory capable of meeting these specifications is presented.

The only storage device, as far as the author knows, which is capable of satisfying these requirements is the Photoscopic Memory developed by the International Telemeter Corporation. The pilot model of this device stores 30,000,000 bits of information with a random access time of 1/20 of a second. The advanced model will store 100,000,000 bits with the same access time.[32] Two of the advanced models of the Photoscopic Memory will store the translation lexicon and provide the desired access time. In the tentative design presented in this report, two Photoscopic Discs will be used for the storage components of the large memory system.

Inclusion of two memories in the translation system at once raises the question as to whether these two memories should be searched in series or in parallel. In other words, when the search procedure for a particular semantic unit is started and the logic has determined by alphabetic considerations in which memory the semantic unit is located, should the machine allow the other memory to remain idle, or should succeeding text words be interrogated for a word whose corresponding entry is in the other memory? Parallel search involves operating the two memories as somewhat independent systems; therefore it is to be expected that more complex and expensive logic will be required for parallel search than will be required for serial search. In order to specify whether series or parallel search is the more desirable, the answers to three questions must be obtained: (1) is the memory search a limiting factor in processing speed; (2) will parallel search significantly improve the processing speed; (3) what additional costs are associated with parallel search?

1. The fact that memory search time is a limiting factor in processing speed may be easily established. The average access time of the Photoscopic Disc is approximately 50 milliseconds. Of the IBM 704 operation codes required for Machine Translation, only the execution of the two logical AND operations requires more than 24 microseconds. These operations require 36 and 48 microseconds respectively; hence with a computer speed comparable to the IBM 704 approximately 2000 program steps can be executed during the time required to locate and read a single entry from the dictionary. In the processing which has been accomplished to date, the longest single processing routine requires approximately 340 computer steps. This routine is the substantive processing program of the first round of processing. These 340 steps include load and punch routines, however. The fact that logical AND is not an IBM 650 operation code contributes to a much larger program than would be necessary for a special-purpose computer. For instance, if logical AND is included as an operation code, and, in addition, if binary machine language with a 36-bit word length is used, the 340 computer steps could be reduced to 24 steps, exclusive of the load and unload operations.

The multiple-form-class processing of this report requires 60 to 100 computer steps, and verb processing 80 to 125 steps, both exclusive of the load and unload routines. The lengths of these programs would also be much smaller for a special-purpose machine.

From these figures it would seem that it will not be necessary to allow nearly as many as 2000 program steps of logical processing for each semantic unit of text material being translated. Even 1000 program steps should be more than adequate. It seems, therefore, that if serial search is performed, dictionary search will be the limiting factor in speed of translation. Even if parallel search is used, it is probable that dictionary search will still be the limiting factor.

2. Ideally a parallel search should double the speed with which entries are located in the dictionary. Actually the speed will not be quite doubled because of the problems associated with coordination of the search in the two memories. In this chapter a design which allows parallel search with almost twice the location speed of a single memory alone, will be presented. This is an attractive improvement and therefore parallel operation is indicated.

3. Parallel search requires more complex circuitry than is required for serial operation, and hence the cost of parallel operation is greater than that for serial operation. As the design which will be presented in this chapter would involve only a modest increase in cost over a serial design, parallel operation is again indicated.

The memory design proposed in this report, therefore, is built around a two-memory system in which the memories are searched in parallel.

A system design which allows parallel search with high location speed and moderate cost must still be

[32] Since this was written, the author has been informed that the capacity of the Photoscopic Memory will actually be increased to about 900 million bits.

presented. The reason for stating the design specifications at this point and the justification later is due to the fact that a discussion of the operation of the memory component must be presented before the logical design can be developed. A brief discussion of the operation of the Photoscopic Disc memory is included in the next section.

Operation of the Photoscopic Disc Memory

The Photoscopic Disc[33] memory consists of a rotating glass disc together with various mechanical, optical, and electronic components. Information is stored photographically on the glass disc in binary coding in 600 concentric rows with 50,000 bits of information in each row.

The memory is self-clocked, i.e., it generates its own clock pulses. The rotation speed is 20 rps, which gives a basic clock rate of (50,000) (20) or 10^6 pulses per sec.

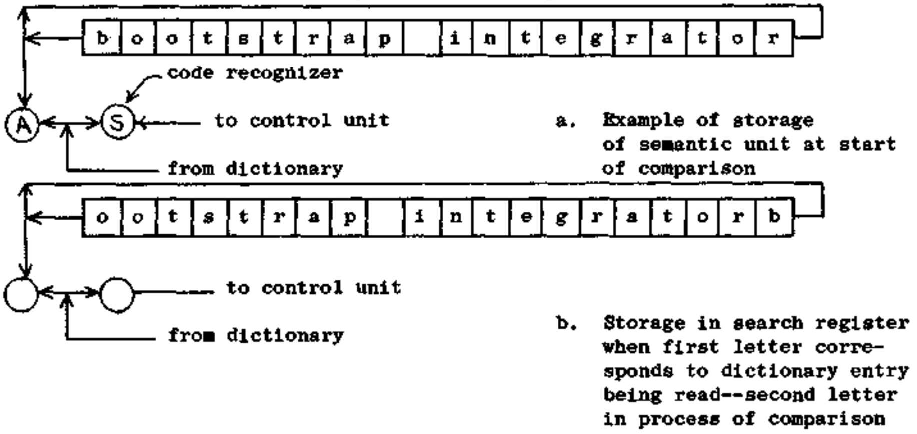
Information is read out of the disc serially by bits. Properly identifying such a continuous stream of bits requires carefully designed logical circuitry of considerable sophistication. In order to make the problems associated with this identification clear, a brief discussion of the search procedure is presented.

In digital memory devices two conventional methods exist for providing unique location of stored information. The first method divides the memory into "cells" and assigns to each cell a unique address. This corresponds to the use of a subject index in an encyclopedia. The reader obtains the page or section number where the subject in which he is interested is discussed. The second method stores identification symbols with each block of information in the memory. Location of a desired block is then accomplished through comparison of an identification sequence with the stored numbers until an exact match is established. This method corresponds to search in a dictionary. The identification sequence (word) is compared with stored words until an exact match is found.

The Photoscopic Disc uses the second of these systems for location of stored information. In order to locate information in this device, a unique 12-bit marking sequence is placed at the beginning of each entry. When this 12-bit sequence is identified, the logic starts comparing the identification sequence, bit by bit, with the ensuing bit pulse train. If an exact match is found, the rest of the entry is read out of the disc. If an exact match is not found, the logic causes the machine to start comparison again at the next marking sequence. An example will aid in clarifying this procedure. For convenience an English word sequence is chosen as the identification sequence. The marking sequence which identifies the source-language part of the entry is $_1$ $_2$. Suppose that the source-language sequence, whose corresponding entry is to be located, is "bootstrap integrator." The entry in the dictionary will be as follows:

α_1 α_2 bootstrap integrator α_1 α_2 cathode follower feedback integrator α_1 α_2

An idealized version of the search circuitry is shown in Figure 4.



a. Example of storage of semantic unit at start of comparison

b. Storage in search register when first letter corresponds to dictionary entry being read--second letter in process of comparison

Figure 4. IDEALIZED DRAWING OF SEARCH REGISTER IN PRESENT INTERNATIONAL TELEMETER DESIGN

Suppose that in the circuit of Figure 4 the "condition o" denotes that state where the circuit is waiting for an α_1 α_2 sequence so it may "start comparing." When the code recognizer identifies an α_1 α_2 12-bit sequence, a pulse is transmitted which sets the logic in "condition 1," the condition where comparison is made bit by bit between the sequence stored in the search register and the bit sequence out of the dictionary. Condition 1 continues until either a noncorrespondence is encountered, whereupon the machine returns to condition o, or until another α_1 α_2 sequence occurs, which advances the logic state to condition 2. During condition 2 the entry is read out of the disc and stored in temporary storage. When the α_1 α_2 sequence is again encountered, the logic returns to condition o.

In the next section the search register is considered in detail.

[33] The Potentialities of a High-Capacity Store for Machine Translation, International Telemeter Technical Report No. 2 - AF 30 (602)1566, August 15, 1956.

The Size of the Search Register

In this section the size of the search register will be specified. The search register is a high-speed store for the portion of the coded text being matched against the source-language portion of the entries in the lexicon. This process is called the "dictionary search."

The search register must have a storage capacity at least equal to the largest source-language semantic unit stored in the dictionary. The length of the longest semantic unit in the present University of Washington lexicon is 39 letters (встречно-ступенчатых направленных зашит) ((of)counter-step-directional-protectors). An examination of an electronics dictionary[34] indicated that about three sequences per page have lengths greater than 40 letters each, and one sequence consists of 49 letters. Many of these sequences could be translated adequately on a word-for-word basis, but others unquestionably are best translated as a single sequence. It would seem therefore, that the search register should have a capacity of at least 49 letters.

Besides the requirement for storage of the longest semantic unit entered in the dictionary, the search register must also be capable of storing a sufficient number of words to allow a high "use factor" for both memories. The "use factor," U, is defined as the probability that a selected memory in a two-memory system will be in the process of search or read-out at any arbitrary instant of time or:

$$U = 1 - \frac{\text{(Probability of one idle memory)}}{2} \qquad (1)$$

For example, if only one word were stored in the search register, only one memory could be searched at one time; and consequently one memory would always be idle. The use factor would then be 50%. If two words were stored in the search register, one memory would be idle only if one of two conditions occurred.

1. The entries for both words are located in the same memory.
2. The first word in the search register takes longer to locate than the second word (each word in a different memory).

The second condition creates a problem because semantic units are not necessarily individual words. For example, suppose that the entry for a particular two-word semantic unit is stored in memory B, while the entry corresponding to the second word of the semantic unit is stored in memory A. If the entry for the second word is located before the entry for the two-word idiomatic sequence, a superfluous entry has been read into temporary storage. This problem is discussed in detail in Chapter 8.

A rough estimate of the use factor of a system with two-word storage in the search register can be determined as follows. The probability that entries for the words in the search register will be located in different memories will be $P = 1/2$. If it were not for the difference in search times for different words, the utilization faction (Equation 1) would be .75.

If the two words in the search register are located in different memories and the first word is located first, then the located word can be read out and a new text word read in. The probability that the entry for this word will be located in the idle memory will be .5. Thus we can say that the use factor of a two-memory, two-word search register system will be greater than .5 and less than .75. These calculations show that a two-word search register does not give a satisfactory utilization factor and therefore need not be considered further.

The search register design requires several address counters. Since the counters will operate in the binary code, it is convenient to make the search register have a letter capacity which is equal to some integral power of two. A capacity of $2^6 = 64$ letters is adequate for idiomatic sequences and is the size assumed in the following calculations. Two calculations must be made in order to determine the use factor of the system with a 64-letter search register: first, the probability of each condition where one memory may be idle, and second, the time in which one memory is idle under each of these conditions. The probability of each condition where one memory will be idle is calculated first.

Probabilities for One Memory Idle

Since a 64-letter register is adequate for idiomatic sequences and will also entirely utilize a six-bit counter, we will investigate first the use factor of a two-memory system with a 64-letter search register. Typical results of a count of the number of words in 15 random text selections of 64 letters each is as follows:

1.	8	6.	6	11.	8
2.	8	7.	9	12.	9
3.	8	8.	8	13.	7
4.	8	9.	8	14.	7
5.	8	10.	8	15.	10

This count would seem to indicate that a 64-letter search register will contain an average of eight words of text. The conditions where a two-memory system with an eight-word search register will have one memory idle are as follows:

1. If the entries corresponding to all the words are located in one search register, one memory will be idle. The probability that this condition exists is:

$$P_{8-0} = 2(1/2)^8 \doteq \frac{1}{128} \doteq .0079$$

[34] Russian-English Electronics Dictionary, U.S. Technical Document TM-30-545.

2. When seven of the eight words have entries in one memory and the eighth is in the other memory, the second memory will be idle if one of two conditions occurs.
   a. The first word in the shift register is the only word of the eight whose corresponding entry is in one memory; and, consequently, the other memory contains the entries for the last seven words in the search register. In addition, the word read into the search register from text when this first word is erased has a corresponding entry which is not in the same memory as the word which was just located. In other words, in this condition, all entries are in the same memory. The probability of this condition is:

$$P_{7-1} = 2(1/2)^8 (1/2) = \frac{1}{256} = .0039$$

   b. The word whose corresponding entry is the only entry of the eight in one of the memories is not the first word in the shift register. In addition, either this single entry must be located before the first word in the search register, or, if the first word in the search register is located first, then the next word read into the search register must have its corresponding entry in the memory in which the entry just located (the first word in the search register) was stored. The probability of either of these two conditions is:

$$P_{7-1} = \frac{2(8) - 2}{128} (1/2) = .055$$

3. The probability that six words have entries in one memory and two words, neither of which is the first word in the search register, have entries in the other memory is:

$$P_{6-2} = \frac{2(7)(6)}{256} = .328$$

The probability that the sum of the search times for these two words is greater than the search times for the first of the six words will not be calculated. A plot of empirically determined data for optimum alphabetic ordering of the dictionary is presented in Figure a of Appendix 3. The linear function:

$$f = a - bx \qquad 0 < x < \frac{a}{b}$$

gives a reasonable fit(see Appendix 3) with the experimental data. Since this function will be used as a probability density function,[35] it is convenient to normalize this function as follows:

$$P = 2 - 2x \qquad 0 < x < 1$$

This generating function for the probability density will be used in the calculations in the following sections of this chapter.

The probabilities of 5 - 3 and 4 - 4 combinations will not be calculated since it can be shown that the probability that a memory being idle because of these two conditions is negligible.

In the previous paragraphs the probabilities for the occurrence of conditions where one memory will be idle were calculated. Even if one of these conditions should occur, one memory would not necessarily be idle for an entire search cycle. In order to determine the probability of an idle memory, therefore, it is necessary to calculate not only the probability of the existence of a condition wherein one memory will be idle, but also the average portion of a search cycle wherein one memory will be idle when each of these conditions occurs. These proportions are calculated in the next section.

Proportion of Time Idle

It is now assumed that the time in which one memory is idle is proportional to the difference in the distance of search in the two memories. This assumption is justified in Appendix 2, where the problem of optimum ordering of the dictionary is discussed together with its significance for devices like the Photoscopic Disc.

For instance when all entries corresponding to words stored in the search register are in one memory (condition 1 of the previous section), one of the memories will be idle during the time that it takes to locate the entry for the first word in the shift register. The average proportional time for this condition (normalized) is:

$$P_1 = \int_0^1 x\,(2 - 2x)dx = \frac{1}{3}$$

For the instance where seven entries are in one memory and one in the other, the calculations are as follows:

Condition 2a is analogous to the selection of an arbitrary y (analogous to the distance searched in order to locate the sole entry in one memory) and an arbitrary x (analogous to the search distance to locate the first word of the seven in the search register whose entries are all in one memory). Then the proportion of time idle will be:

$$\frac{x - y}{x} \qquad x > y$$

[35] The calculations of the next few pages are concerned only with the proportional values (of x and p), not with the absolute values. Normalization, therefore, does not decrease the generality but it does simplify the arithmetic.

and the probability for a particular y and x is:

$$(2 - 2y)(2 - 2x) \; \frac{x - y}{x} \; dxdy$$

and since for an idle memory x y the average time idle will be:

$$P_{t_{2a}} = \int_o^1 (2 - 2y) \int_y^1 (2 - 2x) \; \frac{x - y}{x} \; d\,x\,dy \; \simeq \; .278$$

In condition 2b of the last section two occurrences are considered. For the instance where the single entry is located first, if y is analogous to the search distance for the first word in the search register and x the search distance for the single entry, the average time idle will be:

$$P_{t_{2b_1}} = \int_o^1 (2 - 2y) \int_o^y (2 - 2x) \; \frac{y - x}{y} \; dxdy \; = \; .555 \qquad y > x$$

The second occurrence of condition 2b is identical analytically to condition 2a, therefore

$$P_{t_{2b_2}} \simeq .278$$

For the instance where the 6-2 combination of Condition 3 exists, the search distances for the two entries in one memory may be denoted as x and y respectively. The search distance for the first of the six entries in the other memory is denoted by z. Since (x + y) < z in this condition,

$$P_{t_3} = 8 \int_o^1 (1 - z) \int_o^z (1 - y) \int_o^{z-y} (1 - x) \left(\frac{z-y-x}{z}\right) dxdydz \; = \; .0048$$

For these conditions, then, the probability that a memory will be idle at any particular instant of time because of any one particular occurrence is the probability of that condition multiplied by the proportional time idle under that condition.

The probability of an idle memory at any instant of time is then the sum of the probabilities of an idle memory due to all these conditions, or:

$$\begin{aligned} P_T &= P_{80}\, P_{+8\,-\,+0}\, P_{7\,-\,1_a} \left(P_{t_{2_a}}\right) + P_{7\,-\,1_b} \left(P_{t_{2a}} + P_{t_{2b_2}}\right) \\ &+ P_{6\,-\,2}\, P_{t_3} \\ &= \frac{1}{128}\;\frac{1}{3} + \frac{1}{256}\;(.278) + .055\;(.555 + .278) + .328\;(.0048) \\ &= .0026 + .0011 + .046 + .0016 \\ &= .0513 \end{aligned}$$

The 5-3 and 4-4 conditions are not calculated because the probabilities that three or four randomly selected entries will be located before a single entry also randomly selected, is negligible.

The use factor, U, would then be:

$$U = 1 - \frac{.0513}{2} = .974$$

This system, therefore, produces a random access time to stored information of almost half that of a single memory. Two Photoscopic Discs, with a random access time of $\frac{1}{20}$ seconds each would give an access time of

$$T_{\text{access}} = \frac{1}{40\;(.974)} = .0257 \text{ seconds}$$

in a two-disc memory with 8 words of text available to the search system. A 64-letter search register, therefore, is adequate since it allows almost complete utilization of a two-memory system. A 64-letter search register is therefore indicated.

The Search Register Counters

If the search register is to allow simultaneous and continuous search of both memories and also allow proper look-up of idiomatic sequences, counters are required in order to control the operations of erase, read-from-tape, memory-1 search position, and memory-2 search position.

Usually register storage of the type required for the search register is accomplished with a high-speed flip-flop register. The conventional shift register allows one entrance and one exit from the register. In the two-memory, simultaneous search system which is assumed here, there is still only one source of information for the search register; but two different outputs must be provided, one for each of the two memories. These outputs generally will not involve the same flip-flop at an arbitrary instant of time, but they may. For this reason the shift provision is not particularly useful, and a random access scheme similar to that used with core matrices is more convenient. Several counters are required for addressing the register for such a scheme. These counters will be discussed in this section.

For the particular case of the IT memory, individual letters and symbols are coded with a six-bit code. The tape input may be designed to allow letter-by-letter read-in, so the search register may be loaded on a letter-by-letter basis. Thus, no counter is required to address bit positions within the letter code, only to address the sixty-four-letter code positions in the search register, as described in page 403.

When the search register circuitry has established correspondence between a text sequence and the source-language part of a dictionary entry, the dictionary read station is starting to read the target language equivalents of the entry; and the target-language equivalents are the first part of the entry which will be transferred to the core memory. Occasionally it is necessary to have the source-language part of the entry available to the processing programs; therefore the source-language portion should be stored with the entry in the core. After correlation has been established, the source-language part of the entry is still located in the search register. After the target-language equivalents, tag and perhaps individual-entry subroutine have been transferred to core, the source-language part of the entry could be transferred from the search register to core, and the search register released for new text material.

As soon as the entry or entries corresponding to a semantic unit in the search register have been properly located and read out of the dictionary, the corresponding text words should be erased immediately from the search register and new text material read into those positions. In order that this be done, three counters, all associated with the search register, must be provided to keep track of the following locations:

1. The position in the search register where memory 1 is <u>starting</u> its search procedure. Call this counter S1.
2. The position in the search register where memory 2 is <u>starting</u> its search procedure. Call this counter S2.
3. The position in the search register where the next text word is to be loaded. Call this counter 1.

The load control of the search register could then proceed as follows (see Figure 5.) The load counter, L, is continually compared with the two search counters S1 and S2. When the contents of <u>both</u> S1 and S2 are different from those of L, a read command is initiated, the next code sequence is read off the tape into the position in the search register addressed by the contents of L, then L is stepped up one count. If the new contents of L do not correspond to the contents of either S1 or S2, then another code sequence is read from the tape and stored in the search register address indicated by the new contents of L. If the contents of L again do not correspond to those of either S1 or S2, then the read sequence is repeated until they do. If the contents do correspond, the read sequence is stopped by inactivating the read command and the count pulses to L.

The search register operation involved in the search procedure requires two additional counters, also associated with the search register as follows:

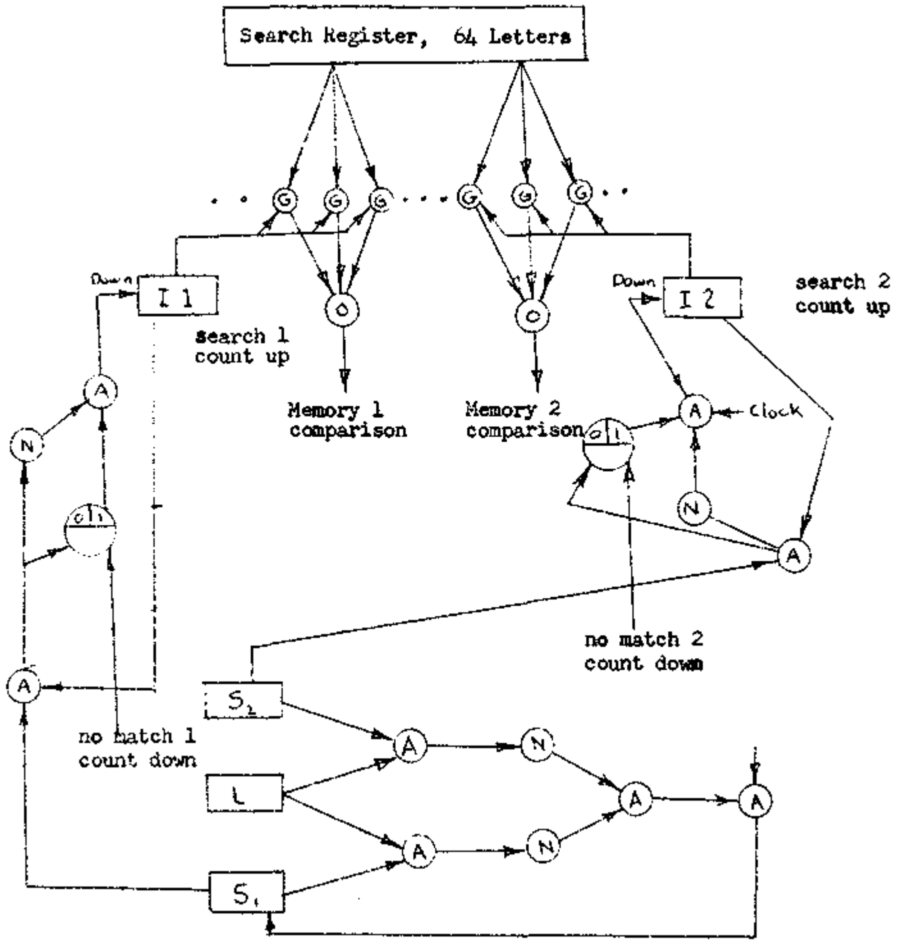
1. A counter to address the instantaneous search position in the search register for memory 1. Call this counter 11.
2. A counter to address the instantaneous search position in the search register for memory 2. Call this counter 12.

If a source-language semantic unit stored in the dictionary has first letters which correspond to the text word sought, but has later letters which do not correspond, then when the first noncorresponding letter is encountered, the machine must cease this comparison and return the I-counter contents to that of the corresponding S counter in readiness for comparison with the next semantic unit in the dictionary. Thus, the 11 and 12 counters must be reversible. The S and L counters, conversely, are only required to count up.

<u>Summary</u>

In this chapter it has been pointed out that parallel search is desirable if a two-memory system built around the Photoscopic Disc Memory is used for the translation lexicon. It has been shown that the design of the logic is not unduly complicated by provision for parallel search.

A 64-letter search register is adequate for temporary storage of coded text during the dictionary search. A shift register is not desirable for the search register since the device must provide outputs to both memories.



FUNCTION OF SEARCH REGISTER COUNTERS

Figure 5

Chapter 8

Control Symbols, Indexing, and Print Out

In the course of translation, the temporary or high-speed memory will store at a particular instant of time the entries for about 115 to 150 text words. Since the lengths of these entries will vary considerably, efficient utilization of the core storage dictates that entries be stored consecutively in the memory rather than by assignment of fixed amounts of storage for each entry. Consecutive storage creates some problems in addressing the stored material, however. In addition, entries are not stored in the high-speed memory strictly according to text order, further increasing the addressing problem. The general problem of addressing entries as they are stored in high-speed storage will be called the indexing problem. Solution of the indexing problem is the subject of this chapter.

In order to index material as it is read from the lexicon into the high-speed storage, the parts of the entries must be identified by stored symbols at the junction points of these parts. These stored symbols will be called control sequences and are the subject of the first section.

The Control Symbols

Control bit-sequences or control sequences are necessary in order to properly identify the entries as they are read out of the dictionary. These control sequences are placed at the junctions of the different parts of the entries so that the boundaries of the adjacent portions of the entries may easily be determined.

Recognition of the control sequences may be accomplished in the dictionary output register by interrogating the proper word positions as they are received from the dictionary. It is convenient that these control symbols always be stored in the same location in the computer word and also that different parts of the entry always start with a full computer word. The one exception to this is the source-language part of the entry. This portion of the entry is not transferred to temporary storage from the Disc, but is regenerated from the search register after the rest of the entry is read out. The source-language part of the entry may therefore be stored in the lexicon without regard for the positions where the break symbols will occur in the computer word. When the source-language part of the entry is regenerated, however, the control symbol must occur in the twelve right-most bits of the computer word. As an example, for the instance of a 42-bit computer word and 6-bit codes for letters, random occurrence of control symbols, in temporary storage, such as the following, should be avoided:

Wd	n - 1	xxxxxxx
	n	ОСНОВАН
	n + 1	ИЯ$\alpha_1\alpha_2$FOU
	n + 2	NDATION
	n + 3	/BASE$\alpha_1\alpha_2$
	n + 4	xxxxxxx
	n + 5	xxxxxxx

Where: $\alpha_1\alpha_2$ denotes the breakpoint or control sequence, between parts of the entry, and xxxx denotes the start of the tag portion of the entry.

Rather, the control symbol should always be stored in a standard position such as in the last six bit positions of the word. The first part of new portions of an entry will then always occur at the beginning of a computer word. For the above example:

Wd	n - 1	xxxxxxx
	n	ОСНОВАН
	n + 1	ИЯ***$\alpha_1\alpha_2$
	n + 2	FOUNDAT
	n + 3	ION/BAS
	n + 4	E****$\alpha_1\alpha_2$
	n + 5	xxxxxxx

Where: * denotes a blank space.

Some additional storage is required in the second case, but the problems associated with read-out to the printer, proper identification of the program steps for the individual entry subroutines, and interrogation of tags are all greatly simplified if the second system is used. The second system is assumed in the following discussion.

Identification of the various parts of the entry can be established by interrogation of the right-most twelve-bit positions by a digit recognition circuit. If a single control sequence[36] were used as in the above discussion, a counter must be used in conjunction with the digit recognizer. The counter would be stepped up one count each time the break symbol was encountered during read-out of an entry from the dictionary. With the counter on "0", the search cycle would be in progress. With the counter on "1", the target language equivalents would be in the process of read-out. With the counter on "2", the tags would be in the process of read-out, and finally, with the counter on "3", the individual entry subroutines would be in the process of read-out. When a control symbol was encountered with the counter on "3", the counter would be returned to "0".

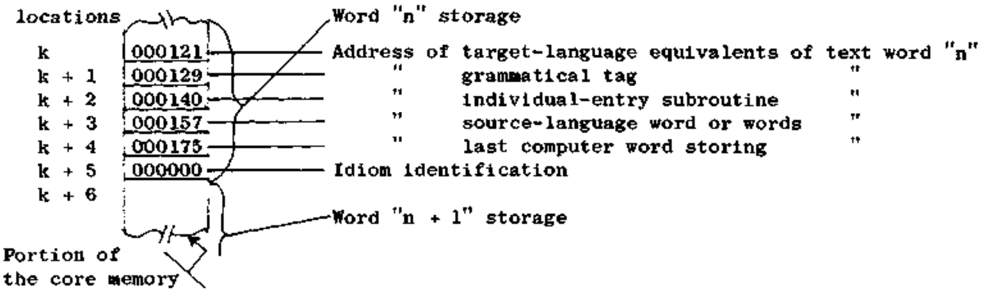
The use of four break symbols requires three more symbols than would be required if a counter were used, and three symbols are a fair portion of the total of 64 unique sequences provided by a 6-bit code. Also the digit recognizer would have to be more sophisticated if four different symbols had to be positively identified. On the other hand, the counter presents a light addition to the circuitry. Since the added circuitry for the counter is not great, a single control symbol is the wiser choice.

In this section it has been shown that the problem of recognition of the various parts of entries may be solved by means of control symbols. The method proposed is similar to that used in the pilot model of the Photoscopic Disc. The differences are due to the fact that the pilot model does not have a clearly defined computer word other than the 6-bit code group for the individual symbols. This proposed design is concerned with a machine which will be optimum for logical processing, and modifications of the design of the pilot model are therefore necessary.

The next section is concerned with optimum storage of the addresses of the entries.

The Index

The index is intended to be a table, located in high-speed storage, where the processing programs may find the address of any portion of any entry which is stored in the high-speed memory, given only the position in the sentence of the text word corresponding to that entry. In this index, six locations are made available for each text word and the memory locations are assigned in text order. As an example of the operation of this index, suppose that the entry corresponding to text word "n" has been read out of the lexicon into high-speed storage. In addition, assume that the locations in high-speed storage where the index for word "n" is to be stored are K, K + 1, . . .K + 5. The addresses of the entry corresponding to word "n", then are to appear in the index as follows:



The six index positions assigned to a single entry store are, respectively, the address of (1) the first memory location storing the target-language equivalents, (2) the first memory location storing the grammatical tag, (3) the first memory location storing individual-entry subroutine, (4) source-language semantic unit, (5) last memory location used to store this entry, (6) idiom identification. As an example, the numbers shown in the index storage locations would indicate: (1) that the target-language equivalents of the entry for word "n" are stored in (high-speed) memory locations 000121 to 000128; (2) the grammatical tag of the entry corresponding to text word "n" is stored in memory locations 000129-000139; (3) the individual-entry subroutine of the entry corresponding to text word "n" is stored in memory locations 000140 to 000158; (4) the source-language words of the entry corresponding to text word "n" are stored in memory locations 000157 to 000175; (5) the last location used for storage of the entry corresponding to text word "n" is memory location 000175; (6) text word "n" is not a part of an idiomatic sequence since the contents of index location "K + 5"are all zeroes.

The first three of the six index locations for an entry are loaded similarly; when a control symbol is encountered, the logical circuitry records the high-speed memory location where the next computer word to be read from the dictionary will be stored. This next computer word will, of course, be the first computer word of the next part of the entry.

Timing of the indexing operations is accomplished with three clock-pulse sequences: one clock-pulse sequence is the one megacycle clock generated by the Photoscopic Disc; the other two clock sequences are generarted from the one megacycle clock by frequency division. One of the two clock sequences is timed to produce pulses at a rate of one pulse for each period of time required to read one computer word out of the Photoscopic

[36] The Photoscopic Disc uses a plurality of control sequences, mainly because segmentation of the bit stream from the Disc is extremely difficult if only a single control sequence is used (cf. Disc Reader High Speed Logic, INTERNATIONAL TELEMETER CORPORATION TECHNICAL REPORT No. 6, November 20, 1956). This problem is extraneous to the discussion of this chapter and therefore is not discussed.

Disc. The other clock sequence is timed to produce three evenly spaced pulses during the period of time required to read one computer word out of the Photoscopic Disc. For convenience, the three clock signals are denoted as follows:

1 microsecond clock sequence	Clock 1
clock providing 3 pulses per computer word	Clock 2
clock providing 1 pulse per computer word	Clock 3

The timing of these last two clock sequences cannot be specified in microseconds since the length of the computer word has not been specified. The length of the computer word will be specified in the next chapter; the repetition rate of clock 3, R_3, will be:

$$R_3 = \frac{10^6}{C}$$

When C = computer word length in bits.

Note that the computer word length must be divisible by 3 in order that each pulse interval of clock 2 may be identical to every other pulse interval of clock 2.

The operation of a representative circuit which will perform the indexing function is shown in Figure 6. Dictionary Output Register #1 receives the output of the Photoscopic Disc in serial form, as explained in Chapter 7. The "match signal" is generated by circuitry which is not shown. This signal occurs whenever a match is established between the source-language portion of a lexical entry and a text semantic unit. The "match signal" persists until the complete entry is read out of the Photoscopic Disc and transferred to store. Since the control sequence is the right-most twelve bits of the computer word, and it is assumed that a match has been established, a pulse of clock 3 occurs simultaneously with the next clock 1 pulse, and consequently at that time the contents of Dictionary Output Register 2 are transferred to high-speed storage and the contents of Dictionary Output Register 1 are transferred to Dictionary Output Register 2. The computer word now stored in Dictionary Output Register 2 contains the control sequence, and therefore is the last computer word of one portion of a lexical entry. The next computer word starts a portion of a lexical entry, and therefore must have an index entry denoting its storage location in high-speed storage. This function is accomplished by the one storage counter, the flip-flops IS_1, IS_2, and IS_3 and the associated circuitry. Briefly the operation is as follows: Flip-flop IS_1 is set to "1" at the same time the contents of Dictionary Output Register 1 is transferred to Dictionary Output Register 2. The control sequence counter is also stepped up one count. The next clock 3 pulse will occur when the next computer word (the first computer word of the next portion of the entry) is transferred to Dictionary Output Register 2. This pulse will set IS_2 to "1" and IS_1 to "0" through A_2. Except for the case of the 5th index position for an entry, the next clock 2 pulse sets IS_3 to "1", which opens A, to the next clock 2 pulse, opening A_4 which transfers the contents of the core-storage counter to the proper index locations. The index gates for controlling storage are not shown.

The first four index locations for each entry are loaded as described above. The last two locations, or the location of the last storage position used for that entry and the idiom identification, must be loaded with different circuits from that described for the first four index positions. In the case of the last location, which is stored in the fifth index position, storage may be accomplished in a way very similar to that described above. The only difference is that the contents of the core-storage counter must be transferred to the index location before it is stepped up rather than after. This is accomplished in the circuit of Figure 1 by the 3 counter. This counter is always set to zero by the clock 3 pulses, then counts up with clock 2 pulses. At count 1 the core-storage counter is stepped up one count, except when the control sequence counter is on "5" in which case the transfer to index is on count "1" and the core-storage counter is stepped up on count "2". In this case, transfer is inhibited by the count 5 output of the control sequence counter. The count 5 output of the control sequence counter sets flip-flop 5 to "1" which opens A_5 to the next clock 2 pulse which then steps up the core-storage counter by 1. Flip-flop 5 sets flip-flop 6 at "1" on the same clock 2 pulse. The next clock 2 pulse opens A_6, transferring the contents of the core-storage counter to the index.

The last index position is reserved for identification of idioms. This problem is considered in the next section.

Storage of Idiomatic Sequence

The idiomatic sequence creates a problem in search and storage in that it is possible for a second or succeeding word of an idiomatic sequence to be located and stored in the core memory before the entire sequence is located and stored. It is because of this possibility that a separate group of 6 computer words must be reserved in the index for each text word (or free form) rather than each semantic unit.

If a word is a non-first word of an idiomatic sequence, this fact is denoted in the index by the occurrence of a unique code sequence in the last index position for that entry. When processing programs refer to text word "n", the first step is to check position 6 of the index. If the unique sequence is stored in this position, the computer treats this index position as if it did not exist.

Two idiom storage counters are required to mechanize the handling of idioms. These are denoted in Figure 1 by ID_1 and ID_2. These counters both count up whenever a space symbol is encountered during the search cycle and are reset to zero each time a new search sequence starts. After the complete entry has been transferred to the core, the three counters perform their duties in idiom indexing by interrogating the idiom counter contents after an entry has been stored in the dictionary. If the contents are non-zero, the index storage counter is stepped up six counts. Since at the instant that the storage of the entry was completed, the index counter was on the sixth index position for that entry, the contents of the index storage counter will be in the sixth index position for the next entry after stepping up six counts. The code "11...1" is then stored in this position, denoting that this entry corresponds to a nonfirst word of an idiomatic sequence. At the same time that the index storage counter is advanced six counts, one of the idiom counters is reduced by one count. That idiom counter is then interrogated for contents of zero. If nonzero, the cycle is repeated. If zero, the idiom indexing is complete; and the second index-storage counter is backed up simultaneously with the second idiom counter in the ratio of six counts of the index-storage counter to one count of the idiom counter. When the idiom counter is on zero, the indexing cycle is complete and search starts for the next word.

The equations for these various conditions are included in Figure 12 along with a schematic diagram of the complete circuit.

The Print-Out Procedure

After processing has been completed, the target-language equivalents must be printed out. The print-out scheme must provide two functions.

1. Location and transferral of the target-language equivalents in the desired text order to the printer.
2. Release of the high-speed memory locations to new entries after the equivalents in these locations have been transferred to the printer.

In the instance where word reordering is accomplished, the index entries are moved about so that at the time print-out is performed the index is ordered according to the desired order of print-out. The problem of locating and transferring the target equivalents to the output printer is thus a relatively simple problem, and can be accomplished by a stored program which is entered when the machine attempts to process a new semantic unit and finds that this semantic unit is an "end of sentence" marker (usually a period).

The target-language equivalents are easy to locate since the first computer word of each index entry is the location of the first word of the target-language equivalent for that entry. The program would start transferring computer words out of the high-speed memory, beginning with the location stored in the first index location of the first word of the sentence. The transfer continues until the position of the first word of the grammatical tag is reached. This location is stored in the third index position for the entry, and the determination is easily made by comparing the location of the word which is about to be read out with the number stored in this third index location. If they are the same, the word about to be read out is a grammatical tag and should not be transferred. The machine would then skip to the location stored in the first computer word of the next index entry, repeating this cycle until the end-of-sentence marker is reached.

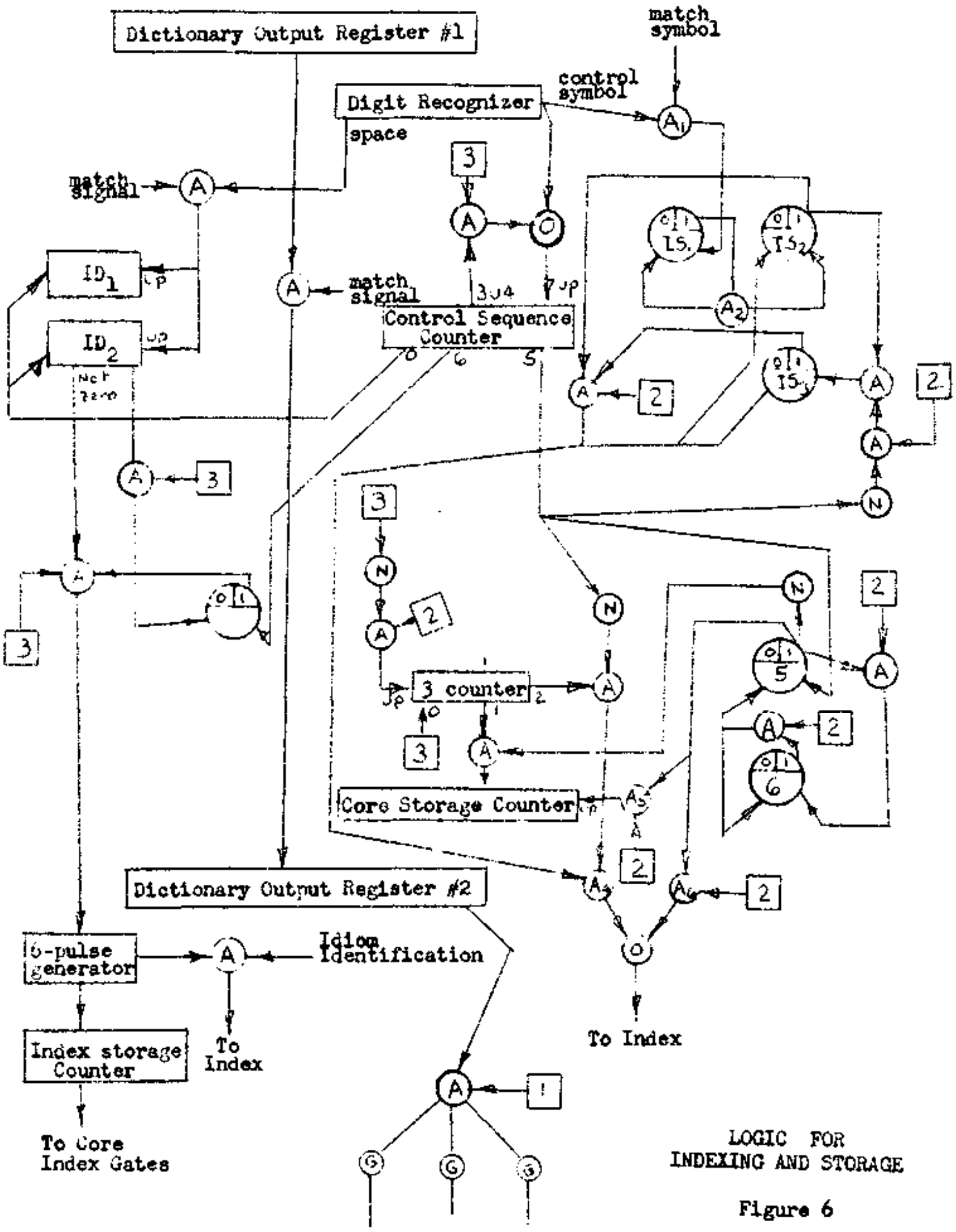
After all target-language equivalents have been transferred to the printer, all storage positions whose contents were transferred to print-out may be erased and new entries stored in these positions. To accomplish this (see Figure 12), a high-speed register is provided in which the last core storage location available to the search configuration is stored. In other words, the next location after the one whose address is stored in this register contains part of an entry belonging to a sentence for which the computer has not yet completed processing. The contents of this register are continually compared with the contents of the storage counter; and, when they are unequal, a "store" command is transmitted to the search configuration. The search configuration then proceeds with dictionary search and transfers material to the core storage until the contents of the storage counter and the register are equal, when the "store" command is removed.

Summary

In this chapter are considered the problems which are associated with indexing material as it is read out of the lexicon into temporary storage, processed, and transferred to the output printer. It has been found necessary to interrupt the access to high-speed storage of the processing programs during transferral of material from the Disc to high-speed storage and during print-out. The disc will be read at 1 megacycle; so the transferral of an average entry of 700 bits would take only .7 milliseconds to transfer. With an average access time of $\frac{1}{40}$ seconds to the two-memory lexicon system, the percentage of time, T, that the high-speed store would be unavailable to the processing programs would be:

$$\frac{.0007}{\frac{1}{40}} \quad (100) = 2.8\%$$

which would not be serious. Print-out would be slower per bit than search, but since only about 15% of the entry is target-language equivalents in the lexicon and this figure should be decreased to 7% or so after processing, this operation should not cause any serious delay. A buffer store between the core and printer should allow read-out to proceed with no delay to the translator operation.



LOGIC FOR INDEXING AND STORAGE

Figure 6

Chapter 9

The Processing Unit

The processing unit of this design corresponds to the arithmetic unit of a general-purpose computer. The logical processing in this unit is performed according to stored programs in the high-speed memory.

The most critical problem related to the processing unit is the selection of the operation codes. The operation code requirements for translation processing are somewhat different from those of a general-purpose computer since the general-purpose machine must be designed for arithmetic operations while the translation computer is concerned only with logical operations. In addition, most of the programs which are written for a general-purpose computer are used only a few times; so the general-purpose computer is designed for ease of programming. The importance of ease of programming is further evidenced by the wide usage enjoyed by interpretive programs such as FORTRAN,[37] BELL LABS,[38] and BACAIC.[39] These aids to programming convenience use a considerable amount of high-speed storage, and the calculation or processing speed of the computer is reduced. For example, the BELL LABS interpretive program for the IBM 650 utilizes all memory locations between 1000 and 1900; these 900 locations are not available to the programmer. In any particular problem some of the subroutines in the BELL LABS routine will be used, of course, but it will be a rare problem which will use most of them. The interpretive part of the routine is a pure convenience which is paid for by a sacrifice of a storage space.

The reduction in processing speed which results from the use of an interpretive program is typically about one-fourth to a little over one-third. For example, if a particular run takes 8 hours on the IBM 650 when programmed with the BELL LABS routine, it may be expected that the same calculations may be performed in about 5 to 6 hours if the programs are carefully written in computer language.

Ease of programming is of primary importance in the translation problem only during the research stage. The commercial translator requires a high processing speed; and, since the processing routines are used over and over during the course of operation of the translator, the routines must be carefully written in computer language.

During the course of designing a computer, the decision as to which operations are to be made available to the programmer as operation codes is made by an evaluation of the utility of the operation against the cost. Thus the decision must be made after a careful evaluation of the problem by programmers and logical designers. The final decision is often somewhat arbitrary.

The programs which have been written for this investigation are sufficient for an evaluation of operation codes for the translation problem.

The Operation Codes

The operation codes which are desired for a translation computer are considerably different from those of a general-purpose machine since a translation computer requires only logical operations to perform the processing programs, plus a few operations which are required in order to move information in, out, and within the machine. The required processing operations are essentially the micro- operations of a general-purpose computer.

A tabulation of the IBM 650 operation codes with a listing of the number of times each code was used in the three rounds of processing and the interpret routine is shown in Figure 1.

IBM 650 Operations Which Are Not Required for Translation

The operations of MULTIPLY, DIVIDE, TABLE LOOKUP, ADD ABSOLUTE or SUBTRACT ABSOLUTE, and RESET AND ADD ABSOLUTE or RESET AND SUBTRACT ABSOLUTE are not required as operation codes. For convenience these operation codes which are not required for translation are marked by asterisks in Figure 1.

[37] FORTRAN, International Business Machine Form No. 32-0306-1, March 1958.

[38] Walontis, V. M., A Complete Floating-Point Interpretive System for the IBM 650 Magnetic Drum Calculator, IBM TECHNICAL NEWSLETTER, No. 11, March 1956.

[39] Grems, Mandalay and R. E. Porter, A Truly Automatic Computing System, PROCEEDINGS OF THE WESTERN JOINT COMPUTER CONFERENCE, AIEE Special Publication T-85.

Figure 1

Operation Codes Used in the Three Rounds of Processing

Operation	Op code	Times Used
No operation	00	26
Stop	01	0
Add upper	10	257
Subtract upper	11	23
Divide	14*	0
Add lower	15	119
Subtract lower	16	18
Add absolute	17*	0
Subtract absolute	18*	0
Multiply	19*	0
Store lower	20	51
Store upper	21	185
Store data address	22	19
Store instruction address	23	0
Store distributor	24	133
Shift right	30	74
Shift and round	31*	0
Shift left	35	103
Shift and count	36	0
Branch non-zero upper	44	117
Branch non-zero	45	6
Branch minus	46	43
Branch overflow	47	4
Reset and upper	60	586
Reset subtract upper	61	8
Divide - reset upper	64*	0
Reset and add lower	65	121
Reset subtract lower	66	0
Reset add absolute	67*	0
Reset subtract absolute	68*	0
Load distributor	69	204
Read	70	6
Punch	71	7
Table lookup	84*	0
Branch on Distributor 8	90-99	144

The MULTIPLY, DIVIDE, and TABLE LOOKUP Operations

The MULTIPLY, DIVIDE, and TABLE LOOKUP operations are not required in the translation process, as is shown in Figure 1. These three operations all require fairly complex circuitry. For instance, a serial MULTIPLY circuit may require 3 inhibitors, 5 flip-flops, 2 6-decision-element counters, and 7 diodes. Other designs are possible, of course, but all are complex. An appreciable simplification in circuitry is thus possible if these operations are not provided.

The Absolute Operations

The ABSOLUTE operations are not required in the translation process. The complexity of the circuitry required for mechanization of the absolute operations, Op Codes 17, 18, 67, and 68, depends largely upon the way the normal ADD and SUBTRACT operations are performed. If complementary arithmetic is used, all negative numbers must be complemented. If the ADD and SUBTRACT operations are performed serially by interrogating the signs of the two numbers first, and if ADD or SUBTRACT is performed by conventional adders or subtractors, the absolute operations can be mechanized by a relatively simple modification of the circuitry for sign interrogate. The only additional circuitry acquired will be that which is required to allow one of the two participating numbers to be always considered positive regardless of the actual sign of the number. The ABSOLUTE operations are not required in the translation process as shown in Figure 1, but the savings are not large.

Shift and Round

The SHIFT AND ROUND and SHIFT AND COUNT operations are not required for the translation process. Since these operations are similar to the normal SHIFT RIGHT and SHIFT LEFT operations, discussion of SHIFT AND ROUND and SHIFT AND COUNT will be included in the next section with the discussion of the normal SHIFT operations.

Required Operations

The Shift Operations

The SHIFT AND ROUND operation is relatively easy to mechanize in the instance of pure binary operation. Probably the easiest way to mechanize this operation would be to add the contents of the rightmost bit position to the rest of the accumulator contents just before the final shift operation.

The SHIFT operations were indispensable for the programs of this report. The two shift operations, SHIFT LEFT and SHIFT RIGHT, accounted for one hundred seventy-nine of the instructions tabulated in Figure 1. The shift operations are of lesser importance in a machine in which the logical AND and logical OR operation codes are included, as will be discussed in detail under logical operations. The SHIFT operations are important in any event as may readily be seen in Figure 1 and should be included in a translation machine. Since the programs are concerned only with logical processing, the SHIFT AND ROUND operation is of no value, and SHIFT AND COUNT is of little enough value so that it does not have to be included.

The Add and Subtract Operations

The ADD and SUBTRACT operations, operation codes 10, 11, 15, 60, 61, 65 and 66 accounted for 1132 of the instructions of the table of Figure 1, and are thus the most used of any of the operations. The design details of circuits to perform the ADD and SUBTRACT operations are discussed in many standard texts[40] and will not be discussed here. These add and subtract operations, surprisingly enough, create a fair amount of circuit complexity.

The ADD and SUBTRACT operations are not required for the program steps where data are actually processed if operation codes AND and OR are provided. ADD and SUBTRACT are often very convenient in instances where the program steps are modified, however. These operations must be included in the translator operation codes.

The Branch Operations

The normal BRANCH operations used in a digital computer require examination of a shift register for one of the following three conditions:

1. contents of register are negative
2. all register positions are zero
3. specified register position is zero

The required circuitry for the mechanization of these operations is not complex, but the number of circuit components required is not inconsiderable. Condition 1, the determination of sign, is relatively simple since this requires the interrogation of only one bit position; and hence only a single AND circuit is required with inputs of a command from the operation code interpreter, the clock, and the minus lead from the sign flip-flop. The output of this AND conveys the required information.

The other two operations are considerably more costly to obtain with practical circuitry. If an ideal

[40] Phister, Montgomery, LOGICAL DESIGN OF DIGITAL COMPUTERS, John Wiley and Sons, 1958, p. 276.

AND were available, Condition 2, all zeros, would be easily obtained by a single ideal AND. The zero lead from each storage element in the shift register would give at least 36 inputs to the AND. Two more inputs would have to be provided: one from the clock and one from the code interpreter. The output of this single, ideal AND would be the desired signal. Unfortunately, the ideal AND does not exist; so the usual practical AND component is restricted to two to four inputs in order to preserve margins for reliable operation. If a very high speed transistor logic were used (no diodes), approximately 17 AND components would be required.

Condition 3 requires the most circuitry of all since only a specified bit position is to be interrogated. The circuit requires at least 36 AND components, one for each bit position. Three inputs to each of these AND components are required: the zero lead from the corresponding bit position storage element, one lead from the clock, and a lead from the operation code interpreter. The output of these AND components would be inputs to a single OR, whose output would be the desired function. Again, no single diode OR will handle 26 inputs and maintain reliable margins; so some additional active elements will be required. Approximately 17 OR circuits, four of them active, will give the desired margins.

The BRANCH operations are essential to the translation process. These operations, operation codes 44, 45, 46, 47 and 90-99, accounted for 314 of the instructions of Figure 1. As was pointed out, a fair amount of circuitry is required for these operations, and also a considerable number of operation codes. For instance, the BRANCH ON 'n' ZERO requires at least 36 operation codes, one for each bit position.

Several other BRANCH operations could be of considerable value. For instance, the programmer is often concerned with the determination of whether a particular word is in the genitive case. If a BRANCH ON POSITIONS 25-30 ZERO operation code were provided, this determination of tag category could be made very rapidly. The large circuit requirements of the BRANCH operations make this prohibitive; and, if logical AND and logical OR operation codes are included, such operations may be conveniently performed in a few program steps without such branch operations. This will be discussed more fully in the section on Logical AND and OR later in this chapter.

The Control Operations

The remaining IBM 650 operations may be considered as control operations since they are concerned with control of the flow of material in, out, and within the computer. These operations are: NO OPERATION (00), STOP(01), the STORE operations (21-24), LOAD DISTRIBUTOR(69), READ (70) and PUNCH (71).

These operations are all necessary in the translation computer. NO OPERATION would be equivalent to the TRANSFER operation, which is required in a single-address machine.

The STOP operation is useful for program check out and should be included. The READ and PUNCH operations are performed in this design by the Search and Print Out configurations, which were discussed previously.

These operations are all traffic operations which require only standard gating circuitry, and, of course, must be included.

In addition to the operation codes which have been discussed, additional codes are required, two of which are logical AND and OR. These operations are discussed in the next section.

New Operations, Logical AND and Logical OR

Two operation codes not provided in the IBM 650 are desirable for a translation machine. Logical AND is of extreme importance, while logical OR is strongly recommended. Both of these operation codes are included in many larger machines, such as the IBM 704 and 709.

The most important operation for a translation computer which is not included in the IBM 650 codes is that of logical AND. The program shown in Figure 1 was written for a machine identical to the IBM 650 except for the AND operation code and a 36-bit word length. With this hypothetical machine an adjective requires 39 program steps for the comparison; while a substantive requires 34, and a particle or an adverb 25. Thus, for this comparison operation, a convenient word length and one additional operation code decreased the number of program steps (and hence the time required to perform the program) by 90%.

The logical AND operation requires almost a trivial modification in the circuitry. Sign has no significance in logical AND (or logical OR). The operation can be performed by the adder by utilizing only the carries output of the adder.

Logical OR also requires almost trivial modification of the standard ADD circuit since logical OR is realized by the sum output of the adder, just as AND is realized by the carries output.

The importance of logical AND extends considerably beyond its importance in determining common grammatical cases. For instance, the programmer often wishes to erase a particular case possibility from a grammatical tag. On the IBM 650 this is a relatively clumsy operation. Probably the best way is to place the tag in both the upper and lower accumulators, then shift left far enough to erase the unwanted case information from the upper, then right far enough to erase it from the lower, then shift back to normal and add the lower to the upper. For example, suppose that the genitive case is to be erased. For convenience, suppose the tag is just 10 bits long and that the third and fourth digit positions (from the left) are genitive case. The program of Figure 7a will then perform the required operation, and the answer will appear in the upper accumulator after program step 6.

Suppose that logical AND is included in an operation code (say operation code 25 is logical AND). The program of Figure 7a may then be used to perform the same operation. In the program of 7b, two instructions are required, one of the two a constant. Note also that six shift operations are required for the program of 7a, and none for that of 7b. As was pointed out in the section on shift operations, the inclusion of logical AND considerably decreases the need for shift operations.

Logical OR is useful for setting to "1" a certain pattern of bit positions in which one or more of the positions may already be "1". This is performed by storing the tag to be modified in the accumulator, then "ORing" with a constant which has zeros everywhere except for those positions which are to be "1", and in these positions the constant contains "1's". If logical OR is not provided as an operation code, each of these tag positions must be interrogated by a BRANCH ON ZERO instruction to determine first whether a 1 is already stored before any attempt is made to add a "1" into this position. Figure 8a shows a program for a single address system with standard IBM 650 operation codes for setting positions 3, 7, and 8 to 1. The particular number which is to be operated upon has 0, 1, and 0 respectively in these positions. Operation code 13 is a skip (or TRANSFER) operation. Sixteen instructions are required for this program, including constants.

Figure 8b shows the same operation performed when the machine is provided with a logical OR operation code. Logical OR is assumed to be operation code 26. In this instance four instructions, including constants, are required.

Other Possible Operations

Other operation codes would be useful, such as branch operations of the type BRANCH ON POSITIONS K-n ZERO. Unfortunately, a considerable number of operation codes are required for such BRANCH operations. As was shown in the previous section, the inclusion of logical AND as an operation code essentially precludes any real need for operations of this type· hence it would not seem wise to include them.

The operation codes suggested will perform the operations necessary for the translation process with circuitry which is modest for a large-scale computer. The program length would have to become of very considerable size before processing speed would be the limiting factor.

The discussion so far in this chapter has been concerned with operations whose utility is evident in a computer of the type of the IBM 650. In the next section it is pointed out that a different accumulator configuration would be useful and that this modification has further significance in the choice of operation codes.

Accumulator Design for the Translation Computer

The selection of operation codes is dependent upon the number of registers used for storage of data in the arithmetic unit. The IBM 650 uses three registers: the upper accumulator, lower accumulator, and distributor. For machine translation two registers are adequate, but some modification to the IBM 650 design for the accumulators must be made. For this design, a two-register accumulator is required. Pertinent specifications of this device are as follows:

1) The two accumulators will be individually addressable, as in the IBM 650.
2) The two accumulators will be tied together for the shift operation, i.e., if a SHIFT LEFT command is executed, the leftmost bit positions of the lower accumulator are shifted into the rightmost bit positions of the upper accumulator. The IBM 650 has this property.
3) The accumulators will be capable of a shift operation performed in a ring, i.e., if a SHIFT LEFT operation is executed, the leftmost bit positions of the upper accumulator, which would be lost in IBM 650 operations, are shifted into the rightmost bit positions of the lower accumulator. This is similar to the ROTATE MQ LEFT operation of the IBM 704, except that in this translation design both accumulators are shifted as a unit and both accumulators may be commanded to shift either right or left. The shift operation as used in the IBM 650 shall also be possible.
4) The BRANCH ON POSITION X ZERO instruction will refer to interrogations of position "X" of the upper accumulator.

The first of these specifications, individual addressability, is provided generally in computers and needs no special comment. The second specification conforms to IBM 650 design. This specification is required for an additional reason in the translator design because according to Specification 4 the instruction for interrogation of bit positions is performed only in the upper accumulator. By the use of ROTATE SHIFT described in Specification 3, bit positions in the lower accumulator can be interrogated by first shifting these positions into the upper accumulator. The properties listed in Specifications 2 and 3 allow the machine to examine conveniently a portion of the tag which is two computer words long. Specification 4 could have applied to the lower accumulator just as well as the upper. This BRANCH ON POSITION "X" ZERO requires a considerable number of operation codes; so allowing "X" to include all positions in both accumulators is not practical.

The only remaining part of the design which must be considered in order to specify the operation codes of the translation computer is the specification of the length of the computer word. This will be accomplished in the next section.

The Length of the Computer Word

In a stored-program machine, the minimum length of the computer word is determined by the requirements of the instructions. For instance, in the translation computer about six bits are required for the operation codes and about fifteen bits for addressing the approximately 32,000 locations in high-speed storage; so the computer word must be at least 21 bits long. Other considerations may dictate that the computer word should be longer than this: redundancy for error detection and error correction, convenient word length for data words, and matching requirements between different memory systems in the computer.

Error detection and error correction are not contemplated in the design presented in this report since the need for such redundant coding can only be established by a careful analysis of the reliability of the system weighed against the complication caused by this redundant coding in the logical design. Since reliability of components is outside the scope of this report, the advisability of inclusion of error correction or error detection codes is not considered.

The minimum convenient length for the data words of the translator is 36 bits (the necessary amount of code for the case information). A longer word length than 36 bits would be convenient in handling tags since positions more widely separated in the tag than 36 bits must often be interrogated in order to establish the occurrence or non-occurrence of a particular condition.

The instructions could utilize the 36-bit word length by the inclusion of more sophisticated programming instructions; but, although such techniques are of tremendous convenience to general-purpose programming, they are not advisable for machine translation. Machine translation programs are special-purpose programs which will be written and stored once, then used continually and essentially without change. Programs for general-purpose computers, on the other hand, are generally used only a few times. The instruction-word format of the general-purpose machine is therefore designed for convenience of the programmer, and storage is willingly sacrificed to accomplish this end. In the instance of the unique programs for machine translation, the programs will be used continually; and ease of programming is secondary to processing speed and the requirements for storage of the programs. The instruction word should, therefore, have a length of about 21 bits.

The 36-bit requirement for data is a minimum requirement. A longer word length would be acceptable or even desirable. If the standard computer word length were established as 42 bits, this would be satisfactory for data; but it is just twice the requirement for instruction words. If the standard computer word is established as 42 bits, two instructions may be stored in each computer word. When instructions are executed, the computer would automatically perform the instruction stored in the left-hand 21 bits first, then the instruction stored in the right-hand 21 bits.

The operation codes for a translation machine may now be specified and listed in Figure 8.

Figure 8.

Operation	Op Code
No operation	1
Stop	2
Add Upper	3
Subtract Upper	4
Add lower	5
Subtract lower	6
Store lower	7
Store upper	8
Store data address	9 (uses lower accumulator for function of distributor in IBM 650)
Shift right	10
Rotate shift right	11
Shift left	12
Branch Zero Upper	13
Branch Zero	14
Branch minus	15
Reset and add upper	16
Reset and subtract upper	17
Reset and add lower	18
Punch	19
Transfer	20
AND	21
OR	22
Branch position x zero	23-64

Summary

In this chapter the requirements of instruction-word and data-word lengths have been reconciled by the specification of a computer-word length of 42 bits. The instruction words would then be stored two to a computer word. The left instruction word in each computer word would automatically be executed first by the computer, then the right computer word.

The processing requirements of machine translation can then be accomplished by 64 operation codes. Fifteen bits are allowed for addressing, representing $2^{15}(42) = 1,376,256$ bits of high-speed storage. With about 105,000 bits reserved for entry storage, about 1,271,256 bits of storage are available for processing programs, or about $\frac{1,271,256}{21} \simeq 60,500$ program steps or just the upper requirement specified in Chapter 6.

IBM 650 Program Sheet

Figure 9a
Erase Payments

Problem ______________________________ Page ________ of ___

Loc.	Op.	Data	Inst.	Upper 8003	Lower 8002	Distr. 8001	Remarks
0001	START			11 11 11 11 11	11 11 11 11 11		Positions 7 and 8
2	35	0004		11 11 11 11 11	11 11 11 00 00		of UA are to be replaced
3	30	0008		00 00 00 00 11	11 11 11 11 11		with zero if one, and to
4	30	0004		00 00 00 00 00	00 11 11 11 11		remain zero if zero.
5	35	0008		00 00 11 11 11	11 00 00 00 00		
6	10	8002		11 00 11 11 11	11 00 00 00 00		

Figure 9b

0001	START			11 11 11 11 11			Use of logical AND to
2	25	000C		11 00 11 11 11		11 00 11 11 11	to accomplish same effect
C	11	0011	11 11				as 9a

Figure 10a
"Set to 1" Program

0001	START			1101011010		1101011010	
2	93	0005					Pos 3 not zero
3	97	0008					Pos 7 not zero
4	98	0012					
4	13	out		1101011 110		"	
5	10	0014				1101011110	
7	69	8003					Program for setting
8	13	0003					positions 3, 7, and C
9	10	0015					to '1' when these
10	69	8003					positions may or may
11	13	0004					not be '1' before
12	10	0016		1111011110			processing
13	13	out					
14	00	0000	0100				
15	00	0100	0000				
16	00	1000	0000				

Figure 10b

0001	START			1101011010			Use of logical OR
2	26	0004		1111011110			to accomplish same
3	13	out					effect as 10b
4	00	1100	0100	1111011110			

Chapter 10

Summary and Conclusions

This report has been concerned with an analysis of the problem of machine translation of scientific Russian into English. The fundamental problems of translation have been considered and an attempt made to develop the relationships between these problems as well as to indicate methods of solution. In particular, the following points have been established:

1. The scientific vocabulary is rich and extensive. Machine translation of scientific material requires a lexicon of considerable size--probably 130 to 182 million bits for translation limited to one field of science.
2. The computer can process material only according to syntactic cues; all semantic and pragmatic relationships which are to be considered in the processing operations must be reduced to syntactic relationships by the programmer and linguistic consultants before the machine algorithms can be written.
3. A means of measuring the quality of translations is of considerable importance in machine translation. An objective measure of the completeness of the translation may readily be defined; a measure of accuracy or correctness, conversely, is subjective by nature, and cannot be objectively defined. An objective measure has been suggested which is adequate for multiple-meaning problems.
4. A shorthand notation for the linguistic specification of the processing cues is of considerable importance for construction of the computer algorithms. A reasonably comprehensive system has been presented. This system is capable of extension to accommodate additional cues as they are specified.
5. General-purpose computers may be used for research in processing techniques, but such machines are not satisfactory for commercial translation. An extensive study of processing using the IBM 650 general-purpose computer has been reported in this report.
6. A special-purpose computer which is capable of producing machine translations economically could be constructed. Only two components of such a machine are not now available on a production basis: the large memory and the print reader. Components which will satisfactorily perform these two operations are in the development stage, however. The Photoscopic Disc store will adequately satisfy the requirements of the large memory, and several firms have reported good progress in the development of an electronic reader. A tentative general design for a special-purpose translating machine is presented in this report. This special-purpose machine would be of considerable importance for research in machine translation and could be used for commerical translation.

   The special-purpose computer suggested is capable of considerable flexibility. It is anticipated that ultimate design requirements will, in some cases, be different from those specified in Chapter 6. In particular, it should be possible to extend the large memory to 3, 4, or 5 Photoscopic Discs if such a large lexicon is found necessary. It is pointless at this time to argue that the lexicon should be of a capacity of, say, 300,000,000 bits rather than 200,000,000 bits. Our MT operational lexicon will approximately require a 30,000,000 bit capacity. It is important that the translating machine be capable of extension as larger lexicons become available.
7. The foremost problem of machine translation is not a problem of components; the components are much nearer to realization than the knowledge of how they may be used. Linguistic analysis of the translation problem still must advance considerably before satisfactory machine translations are possible.

   The linguistic cues which are required are manifold, complex, and can only be specified by a concentrated effort by well integrated teams of technically competent personnel.

In conclusion, it is the belief of the author that research in machine translation will supply knowledge about the fundamental problems of language that will transcend applications to machine translation alone. Such fundamental problems as automatic library search and bandwidth compression of communications should receive partial solutions from research in machine translation.

Appendix I

Probability of High-Frequency Words

The probability of occurrence of individual members of the word count is of interest. Epstein[12] points out that it is reasonable to consider the probabilities of all words which occur ten or more times in a sample. Ten occurrences are chosen since the probability is then established that in a large number of samples, the occurrences of the word will, for nine-tenths of the samples, be closer to ten than twenty. In other words, if a word occurred ten times in a sample of 100,000 words, the probability would be 1/10,000 for that sample. Confidence level 0.9 in this instance means that if ten of these 100,000-word samples were taken, the expectation would be that in only one of the ten samples would the number of occurrences of that particular word be closer to twenty than to ten.

A more precise and somewhat more useful way to consider this problem is to develop an expression for the range of probability which a word may have with confidence level 0.9. Specifically, if ten samples of the same size were taken, and the number of occurrences of a particular word in each sample was divided by the number of words in each sample, then it may be expected that these ratios will be of the same order of magnitude but certainly not equal. An equation will now be derived which expresses the increment ϵ by which the ratio of the number of occurrences of a particular word divided by the number of words in the sample will differ from the actual probability of the word in an infinite sample. Confidence level 0.9 is assumed. In other words, on the basis of ten samples, only one ratio may be expected to differ from the actual probability by more than the increment ϵ .

For any $\epsilon > 0$, the event

$$\left| \frac{S_n}{n} - p \right| < \epsilon \qquad \text{is the same as} \qquad \left| S_n - np \right| < \epsilon\, n \tag{1}$$

Where: *S_n is the number of occurrences of the word in the sample*

n is the sample size

p is the actual probability

Dividing (1) by $(npq)^{1\ 2}$, where $q = 1 - p$, we get

$$\left| \frac{S_n - np}{(npq)^{1/2}} \right| \epsilon \; \frac{n}{pq}^{\;1/2} \tag{2}$$

We may also write

$$\left| \frac{S_n - np}{(npq)^{1/2}} \right| = S_n^* \; ,$$

Where S_n^* is the reduced number of successes. The De Moivre-Laplace limit theorem states that

$$\Pr\,(a < S_n^* < b) = \Phi\,(b) - \Phi\,(a)$$

where Φ is the normal distribution function defined by

$$\Phi(x) = \frac{1}{(2\pi)}\ 1/2 \int_{-\infty}^{x} e^{-\frac{y^2}{2}}\, dy$$

$$\text{If} \quad a = \left| \frac{S_n - np}{(npq)^{1/2}} \right| + \epsilon \left(\frac{n}{(pq)} \right)^{1/2}$$

Then

$$\Pr(a < S_n^* < b) = \Pr \left\{ \left| \frac{S_n - np}{(npq)^{1/2}} \right| < \epsilon \left(\frac{n}{pq} \right)^{1/2} + \Phi \left[\epsilon \left(\frac{n}{pq} \right)^{1/2} \right] - \Phi \left\{ \left(\frac{n}{pq} \right)^{1/2} \right\} ,\right.$$

where Pr is the probability of the bracketed expression

This can be shown to give

$$\Pr \left\{ \left| \frac{S_n - np}{(npq)^{1/2}} \right| < \epsilon \left(\frac{n}{pq} \right)^{1/2} \right. = 2\,\Phi \left[\epsilon \left(\frac{n}{pq} \right)^{1/2} \right] - 1$$

on the basis of Eq (1)

$$\Pr\left\{ \left| \frac{S_n}{n} - p \right| < \varepsilon \right\} = 2\Phi\left[\left(\frac{n}{pq}\right)^{1/2}\right] - 1 \qquad (3)$$

For a confidence level of 0.9

$$\Pr\left\{ \left| \frac{S_n}{n} - p \right| < \varepsilon \right\} = 0.9 \; ;$$

so

$$2\,\Phi\left\{\varepsilon\left(\frac{n}{pq}\right)^{1/2}\right\} = 1.9;$$

and

$$\varepsilon\left(\frac{n}{pq}\right)^{1/2} = 1.65 \; .$$

For a sample size of 30,000,

$$\varepsilon = \frac{1.65}{\sqrt{30,000}} \sqrt{pq} = .0095 \sqrt{pq} \; ,$$

and since

$$p \ll 1 \; , \quad q \approx 1 \; ,$$

then

$$\varepsilon = .0095 \sqrt{p} \; .$$

The percentage increase in the probability of a particular member in the scientific word count over the probability of the same member in the general-language word count is defined as

$$A_1 = \frac{(p_s - \varepsilon_s) - (p_g + \varepsilon_g)}{p_g} \qquad p_s > p_g \quad \text{and}$$

$$A_2 = \frac{(p_s + \varepsilon_s) - (p_g - \varepsilon_g)}{p_g} \qquad p_s < p_g \; , \qquad (4)$$

where A = percentage gain,
p_s = probability of the word in the scientific word count,
p_g = probability of the word in the general word count,
ε_s = range of probability of the word in the scientific word count for confidence level 0.9,
ε_g = range of probability of the word in the general word count for confidence level 0.9

Example 1:

The Russian word И occurred 1132 times in the scientific word count and 1160 times in the general language word count. The increase in probability is then:

$$P_s = \frac{1132}{30,000} = .0377 \qquad P_g = \frac{1160}{29,345} = .0387$$

$$\varepsilon_s = .0095 \sqrt{.0377} = .00185 \qquad \varepsilon_g = .0095 \sqrt{.0387} = .00187$$

$$P_s + \varepsilon_s = .0396 \qquad P_g - \varepsilon_g = .0406$$

$$P_s - \varepsilon_s = .0358 \qquad P_g - \varepsilon_g = .0368$$

Since the ranges overlap $[(p_g > p_s)$ and $(p_s + \varepsilon_s) < (p_g - \varepsilon_g)]$, the increase in probability is zero. This means that it cannot be said that with confidence level 0.9 the probability increased at all.

Example 2:

The Russian preposition в occurred 1076 times in the scientific sample and 724 times in the general language sample. The increase in probability of the Russian word В in the scientific over the general language samples is:

$$P_s = \frac{1076}{30,000} \qquad P_g = \frac{724}{29,345} = .0241$$

$$\epsilon_s = .0095 \quad \sqrt{.0359} = .0018 \qquad\qquad \epsilon_g = .0095 \quad \sqrt{.0241} = .00147$$

$$P_s = \begin{matrix} .0341 \\ .0377 \end{matrix} \qquad\qquad P_g = \begin{matrix} .0256 \\ .0226 \end{matrix}$$

$$A = \frac{.0341 - .0256}{.0241} \quad (100) \quad = + 35.2\%$$

The increase in probability of a word in general-language material over the same word in scientific material may be defined by equations similar to Eqs (4):

$$A_1^1 = \frac{(P_g - \epsilon_g) - (P_s - \epsilon_s)}{P_s} \qquad\qquad P_g > P_s$$

$$A_2^1 = \frac{(P_g + g) - (P_s - \epsilon_s)}{P_s} \qquad\qquad P_g < P_s$$

Appendix II

Print-Out of the Third Round of Processing

A complete print-out of the Third Round of Processing is presented in this appendix. The words were stored consecutively in the machine according to the listing, starting with the address listed in the left most column of each row. For example, the word 60 0006 1161 is stored in location 0160, 65 8002 0169 in location 0161, 000000 0000 in location 0162, 2201170120 in location 0163, and 60 00670221 in location 0164. The first 9 rows of the print-out are concerned with the loading of the program, which accounts for the zeroes in the address column of the print-out.

```
0001 6919531954  2100008000  2400001956
0000 6919531954  1119571952  2419111955
0000 6519531954  6019561950  2419131955
0000 4419531944  1060011954  1019581955
0000 2419151911  3000021917  4619021920
0000 2419211911  2100001928  2119321943
0000 2419271911  6519011916  6019321937
0000 2419331911  1019361941
0000 2419391911  1519291933  6019081918
0002 6000011017  1100010005  1680010011
0009 6001120317  1600220028  1500140019
0016 6900060153  1000200075
0023 6500010117              2400280031
0031 1000340039  6000540359  2400360089
0038 1500416002  1580010047  9304430255
0045 2400480051              2000010004
0052 1000001117
0059 1080010017  1000000055  6080030119
0066 2100030603              6004710275
0073 1000240129  1000000553  1580018003
0080 7109270177  2200350038  6001350129
0087 6180030135  6001410095  2400420045
0094       60000  4400490250  6000030755
0101 4602500405  9310551207
0110 1000000105  3000040171       10003
0117 1000000105  9780020664  3500040029
0124 2009730226  6000540309  6002790233
0131 6501340139              6001590463
0139 1500928002  6001340339
0146 9608990295  6901080661  6912590204
0153 9401030008              3500020111
0160 6000010155  6580020169
0168 6900061161  3000040079  6501590563
0175 6900760081    16340000  6001300235
0182 6002340389  4400370088           1
0189 4600880043  2000001062  1000940099
0196 4608201199  2400721275  6902910144
0204 2405571298  3000020161  6005590713
0211 6000000355  1000940299  2100240064
0218 6900061211  6000540409  1000160073
0225 2001670270  2409830086  6900720475
0233 2407090212              2401360191
0240 6001230727    14680000  2400140717
0247 1503541059  6007090516  2101340057
0254 6004640569  3500040065  9106090361
0262 6003150419  2000540107  6000670321
0269 6000540459  6001230077  2400540157
0276 6980030564  4601800603  6000570301
0283 6901360005          10  5180000005
0290 1000000000  6004200725  6900006657
0297 6003600485  6000981103  2101540507
0304 9806570605  3000040115  1505598002
0311 2003030206  6001380193  2100300025
0318 6500020715  4401730251
0325 6901280181  2401350388  6902300333
```

```
 1619588002 5619576002 2100011955      10000
 6919561957 7019131913 2419448000
 5919561957 1519028003 2419508000
 1560018003 6919038002 6919088002      10000
 1519211924 4419291930 4419231926 3500061935
 1080011939 2019291932 4619281929 6919311938
 2100001928 1980011925 6019021918 6019021918
 6560021915      10000 1119401919 2419321944
 1560011922            2019298003 7019271927
 4400590010 6501590013 1500608002 1502118002
 6001650319 2100160021          CR4400690070
*1500228003      10000 2400240027 2400000003
 6501590363 2400300033          CR4400830084
 6500008002 1000000205          CR2001790022
 6500010035           *6500460139 6000970101
          CR2001410044 2400530006 2400540007
*3000040015 1580030663 6901100163
 1002150253 2100160070 6500670071 4402190170*
 6000540209 6501590113 2400540007
 6001790185 4600760131 1000000205 4500320133*
 6000540259 6001590263 1000360091 6902390142
 6001340689 1000420053 6900060353 2104710124
          CR          *2101080511 2401030056
*2409270080 3500020061 1005098003 1001608003
 2100540057 6500670121 4402690220 6004710575
 1500238002 2400540057    9560000
 2201858001 6900000403 1000300055   13000000*
          CR          *1500200675 1000940349
*2402450148 1580010351 2405570415 2400960151
 2100460199 2402030106 1000940249 6000460801
 2002610364 1500200175 1000200253
*2201170120 6000670221          CR6905190172*
 4401250026 2402450196 2001650066      90000
 6180030379           *6005590863 2201658001
          CR6903390242 1000201175 6002410145
 6001230277 1560010401          CR2101230126
 6180030457 2402530156 6960030706 6000721077
 1002108003 1004110253 4400630054 6000020235
 6500670271      40000 6004710775 6503200139
 2400540107 1000000000 2409281297 2405590562
    9600000 6501320139    9290000 1101840453
 6005760831 2400460499          CR6901380341
 1000968003           *         CR6000000693
 6003030307 4602390505 6005440302
 1502608002 1004618003 4402130114 1000030305*
 2104200273 9005000550 6580020375 6900071261
 6000011115 6001760331 6007761133 4602610229
   15510000 6001061113 6012101032    9481600
 6005761031 6904200473 7109770477 4601920603
 1560010451 6001340789 6000460651 6000060793
 2404530306 4602540555 1008568003
 4403110012 9611111207 4403130164           *
 6500670371   17170000 6000070655 2403700223*
*2400540207 6900070873 6180030381 1008288003
 5000060622 6901820335 2100000116 2402240137
```

```
0332  6001340990  2400140467         100
0339     9770000  1010798003  2401940147
0346  6000000893  6007501005  6001350640
0353  9202560158       50000  1100580103
0361  1004140253     9700000  2100540157
0368  2000020236  2100970400
0375  3500010231  6902340337     1090009
0382  6001341190  2400140667  6007090665
0389  1580010247  1009438003  6900460449
0396  1009438003  1011001105  6008190582
0403  9003570208  6900060949  6501080139
0410  9109630312       30000     9800000
0417  6004700625  6000010865  2400720425
0424  4408781028  1000940399  2400140817
0431  7109270377  6000190482
0438  6900061093  1506730527  3500091262
0445  4411491150  6000980854  1507090868
0452  6000970502              6900070999
0459  4406130314  1510648002  6500050455
0466  3500081050  6905200025  2100010286
0473  2401230376  6000981254  2400140367
0480  3000000000  6001380293  1008358003
0487  6005760631  6002440699  2102440297
0494  6000461051  6000980754  6000050804
0501  4408050239  4606780906  9804530258
0508  6003100472  6900060203  1012158003
0515  2400140567  1009198003  6905550025
0522  9704580810  2401540807  6008190574
0529  6000000855  9706330585  7109270280
0536  4609000792  2103800583  6004711276
0543  6001350196              6006081072
0550  6080010707  2403101213     9480652
0558  2108190722              9206031265
0565  6001180723  1009691023  6905550025
0572  1003288003  6900721225  1011288003
0579  6002440549  9606830685  1001840539
0586  2108190636  2106190800
0593  6000460851  9706470545  6005441052
0600  6900980551  6901540757  6008190624
0607  4601160661              1500200325
0615  6005760781  1010198003  9105700619
0622  9104580378  6904710224  1011788003
0629  6002820587  9802330683  1002840589
0636  6002340842  1006190923  6007091232
0643  1002468003  6000460402  4411021149
0650  6007030907  1004048003  6001351140
0657  9204100312  2106080524  6900070503
0664  6009670621  1003188003  3500090445
0672  4404280603              6003100724
0679  3000090849  9606830735  1080020639
0686     9480736  9500400650  6001231283
0693  3000040654  2106080274  6000460658
0700  9307530905  1004548003  2108190674
0707  9907610462  2106080524
0714  6005760581  3000060826  1010598003
```

```
 2402380291  6003200340  2407090562  2400970600
 6900070737  1508128003  6007090324  1507090915
 2103200323  2405530356  1503540809  6001350890
 6507090763  9903640062  9411620603  4404130214
 6001670421  6009530607  6000070705  6905570025
*2400540257  6007091209  2409280531  3500060940
 6001361143  2100140417            LN2105590262
 2403200623             *6901320385  6907090512
 6900060787  6005760731  6000981003  1007098003
 2101320285  6006530407  1503540859  2105440697
 6000001053  2403100813  6000070904  4405130264
 2100360133       20000  6909610514  1008698003
           LN4402250076  6003201026  2409850288
 4502800481  1004441250  6002320437  6001320537
*6007090815  2104710238     9480486  4606030441
 6001541109  6900060837  9701460295           7
 6006841141  2401060911  6009570611  1503541009
 3500050267  6003100966  2101350188  6001340740
 2009730276  2000540207     9530000  2400140517
 1501220227     9510000            LN1012778003
 1505298002  6004330187  6000030822  2004330186
 6002340485  6007090965  1004380343  6005590536
 6000061187  6000460751  6000071189  4403970246
 4406830202  6008191134  6901230326  9903040605
 6900061049  6001060913  6000061153  6180030265
 6180030369  2400960601  2100420170  2402380291
 2100011823  6001350289  6003700825  6900740127
 4601780329  2103800483  1503308002  2100030603
 6000000992  4404870488  6001340891  4410390240
 2105760764  1009938003  6003100618  6000721258
 6005440910  1004440201  6010850642  1007738003
 1000008002  6900071099  6001380243  6000001253
 6500001015  6903200523  2100540257  9206170619
 6080020677  1501220327  6001081063  1501740479
 4601780379             *2102440347     9480352
 1008858003  6000720577  6000980634  6001231177
*2105760486  9110430495  3500091212  4609000640
 6005441834  6000961104  6004360742  3500041159
 6000970251  6000060901  6001380143  9109001000
 6003101216  4403650116  2404200573  2100480220
 1010218003  2009830555  6003700925  2105760579
 1502281283  1502440799  1507091165
 6003100682  6007090616  2105440448  3000091155*
 4403930294  4604560494  1002841269  2106080734
 6000451184  9302520588  6006860892  1000028003
   13840900  4409000820  9007590312  6003101220
 1009186003  6004710525  6901940197  1000168003
 6906200125  3000040829  1602220427  6003701075
 2004711299  4606290430  1104800535  6001231086
 2105441247  6001541260     9481048  6001231180
 2105760294  2108190722  9606830396  6001081235
 6001080746  6003101172  6010360992  1009128003
 9907000654  4409000820  9009090619  9007601170
*1009681173  4404150166  6900060857  4600120217
 6906700025  4410710820  2401320435  1604140769
```

```
0721  1080020729  6007090772  2106090862
0728     9480452  6080030637  6001321137
0735  6002340390  6004710786  9700900295
0742  2106080734  1002968003  9308470608
0749  3000081067  9906290664  1005048003
0756  6001080572  2401340287  2106080524
0763  1500660521  6002440649  1003688003
0770        1000  3000080881  1004788003
0777  4409001205     9481293  6080030987
0784  1007888003  1080021243  4609000942
0791  6005441251  6003200586  3000040629
0798  6006191288  1500521007  6007090864
0805  1080011113  9007601170  6905590612
0813  1000940149  6580031061  1004188003
0820  6007090660  3000020627  1580031179
0827  6000981203  6000050374  4409000620
0834  1008368003  6000060941  6000720686
0841  4611490695  2105440648  1003468003
0848  6011451195  1580011157  6008031057
0855  1001840439  6000070840  9010110762
0862  6005591263  4602160603  1011678003
0869  6000040466  6000061166  1080020779
0876  3500040937  9206830542  6001231078
0883  6001320396  3500011241  6000070991
0890  4606780644  2106191230  2106080734
0897  6007520708  6006960946  6001340839
0904  9704560348  6004710676  6001081272
0911  6901350338  6000010749  6902660150
0918  6000040710  6000050566  6009700793
0925  1000201025  6001380743
0937  1001900395  2106190836  1004428003
0944  2105440281  6000460995  2106080486
0961  6004640469  6909510104  6001081013
0968    80000000     8000000
0976  1111210733
0987  3000010493  1011928003  3000081620
0994  9304981084  2106191045  6001541046
1001  4608201255     9481252  1004068003
1008     9481058  6902300383  6001351041
1015  3000040876  3000040777  3500020623
1022  6005781033  6980030530  1012788003
1029  4405360900  2106191080  9304841623
1036     9481170  9612020608  3500020645
1043  9304460348  2105440598  6003101095
1050  3000090871  1010048003  1011068003
1057  2106190800  6000971108  6900720426
1064  2100030603  1002728003  4409000620
1071  6001380643  1006288003  6980030580
1078  4609000332  6000071126  6000981158
1085     9481135  4611491042  4405411162
1092  2106191136  9780020664  9010000900
1099  9009030603   700000000  9110540056
1106  6000051156  2106190372  4611490680
1113  1580030571  1001688003  3500020621
1120  3000091267  1500040818  4409000678
```

```
 2105440897 4602450429 1003228003 4606031081
 1003340689 2106080432 1011718003 6008190784
 6001341091 2108190372 2105440597 1004441249
 9303980608 2108190796 6001380694 6012361142
    9480802 6003200726 1012578003 3000010811
 9905550312 6002840690 9805640462 9810110867
 3500080635 6903101163 1011218003 2006730476
 6000020599 2108190824 4602780603 1507090964
 6000061027 9103840236 6001080832 1080020641
 9201400603 6000061191 1002928003 9905940545
 6000460702 6000461060 6003100846 6005520658
 1005548003 6000970852    9481804 3500021120
 2106080924 6903620465 6000971151 6580020669
 1011198003 6907200025 1518180976
 3000040533 6001360844 1000200875 3500060887*
 6000001127 9204340286 2108190882 6005591018
 9004910603 6000040884 1003428003 9712930588
 2105440947 3000091088 2105440898 9706020608
 1006048003 4609001006         20 1004088003
 2108191224 6904120515    9481010 2406140767
 3500080771 1011691073 9610110650 1011718002
 3000021229 9706290905 6000981056 2109280431
 6000061076 6001081030 1580010989 6001380944
 2105440698 1007090765 6000071291 1003928003
 3000040704 9310470608 4611490945    9481149
 6018180739 9912050456    9480760 6001081164
 2108120565 2105441097 9905550619 1005328003
 9412170819 1504688002 4409000543 6003200526
 6001380643 1505288002 1507090460 6008191024
*          CR         *          CR
 3000090424 9907450546 6001320938 6000060440*
 6009020758           *9501020850 6009560711
 1512678002 1005618003 1012698003 8888888888
*          CR         *          CR
*          CR         *          CR          *
 2108190474 9711830608 3000041281 6000060590*
 6010020808    9481110 9700901207 6018191107
 6000061101 2108120297 6001080774 2010610814
 6001540510 6000031259 1003168003 3500020721
 4606830422 6000040666 6000061266 6000000668
 2103700373 1008798003 3500090497
 2106080432 2105880791 4410000900 6000051282*
 6007090716 9807441170 4611830795 6001341092
 1010968003 9710370608            CR9201520603
 3000040916 9305080810 9703081207 2105440997
 2108190632 1009121117 6007091065 1003668003
 6080030626 1012218003 6000040766 3000041181
 3000090645 1000201125 9905361131 1008308003
 6001341139 6000071182 2105880791 6001350592
 4412421292 4411490344 9709000608 2108190584
 2105440848 6000061245 6010081274 6000061148
 6005441152 1005068003 2106081022 6980031112
 1007128003 6000971160 6000460701 9905680248
 1008708003 3500060681 3000041029 6000040866
 4408271199 4409001149 2103700423 9205381131
```

```
1127  3000041237  6000061040  6000070883.
1134  1008888003  6000971185  6000981186
1141  2106080734  2106081286  1004968003
1149     9480683  6001231228  4604580756
1156  3500021074  3000021014  2105441147
1163  2401380391  1002188003  1505188002
1170  6008530558  2000040603  2106081222
1177  4606031231  6000071090  3500091201
1184  2108191234  4605000692  2105440748
1191  9209940596  6000050736  3500021200
1199  6000460851  3000070672  3000090672
1206  6000001256  6001081114     9481110
1213  6901340237  6001081264  6000060560
1220  1012238003  6000001118  6007281063
1227  3000071093  4611490382  3500020785
1234  6003101284  2106191285     9481238
1241  3000091034  1004441138  1008191273
1249  6980030606  6980030806  1011548003
1256  3000041124  6000050804  2105441157
1263  4606030917  1002688003  6007090416
1270  3500021227  6000001070  2105440747
1277  6000070522  6000061240  6000071250
1284  2105440548  6001381044  6005440988
1291  9709001084  6006191188
1298  7009610561  7009500450
1800  6018011802  1000100000  2109511612
1807  1016086003  6000061609  1109511610
1814  2109521803  6016161817  2100060900
1821  1080021822  6080030345  6002341624
1828  1502341829  1518308002  2100031604
1298           9  7009500450
```

```
 2106080924  6002341068              CR2105880791
 1004908003  6980031094  1004928003  4609000794
 6000061194     9481245              CR6012081100
 1012068003  3500091123  6000060790  1101841089
 4407140764  4611830782  9500900850  6000971001
 9909200921  6000030914              CR    800000
 6980030630  1012798003  1100970501  9706830538
 4611830534  4406831219  9006030638
 9906830542  1011448003  9905430394  2108190874*
 9307981238  2106060486    12460900  6008601280
 6008191174  1005568003  3500041122  6000971001
 1010128003     9481183  9212140603  3000090783
 1010208003  1007700776  6003201226  6001540768
 6000050804  6003100908  2402340387  1011298003
 6000981258  6001340540  1010358003  4406880603
 4412190683  6001350895  4606030593  9908940595
 6000071294  6000971295              CR6012080732
 6001350841  3000041066  2105440797  6001081116
 3500090679  1010828003  9006150603  4411681218
 3500091087  2100030819  1012718003  6000001016
 6000031281  2106080924  6106140719  4606830730
 2106080432  4406831183  3500091233  4609000738
 4407410603  1012448003  6080030447  9711490608
*9711490608  4611490696  6000021204  7009620162
*           CR          *           CR
 6001541807  6018051806  1010000000  2109511815*
 1501541811  1509528002  6018131814  2100061000
 2109521803          80          90  3500081821
 1018258003  6000031826  9418271804  1007701828
            CR          *           CR
*           CR          *           CR          *
```

Appendix III

Optimum Ordering of the Dictionary

Input-Output Equipment

Ideally, the input to the translator should be the printed page. Recognition, while of extreme importance, is outside the scope of this report and will not be discussed. Thus it is assumed that the text material has been coded on tape, and that these tapes are used to introduce the text material into the computer.

For the input tape, punched paper tape is more satisfactory than magnetic tape. This is due to the fact that paper tape is read by virtue of its *position* with respect to the reading device, while magnetic tape is read by means of *velocity* of position with respect to the reading device. The tape is read only a few bits at a time, so unless a buffer store is used, magnetic tape would require a far more complicated control than paper tape.

Tape could also be used as the output media, but many excellent high-speed printers are available, so it is possible to produce a printed output directly from the computer. Hence it will be assumed that the computer output must be in a form acceptable to a high-speed printer.

The Large Memory (Dictionary)

Several memory systems are now in existence which will store the dictionary necessary for Machine Translation, even a large one containing every relevant paradigmatic form of the contemporary source language. Since these very large memory systems tend to be serial access devices, or at least semi-serial devices, optimum storage and search techniques for a serial memory will be considered in some detail.

In order to make the development more general and since at least two Photoscopic Discs are required for the dictionary, it is assumed that the translation dictionary is stored on two identical serial-access devices. If only one device is assumed, the design problem would be considerably simpler since the problem of coordinating these memories would not exist. The design based on two devices can be extended easily to apply to more than two devices.

Information Storage on a Serial Access Memory

For a serial access device used for dictionary storage, the average access time will be dependent upon the way the source language words are coded in the dictionary. For minimum access time, the entries must be ordered in such a way that the words which occur most often in text will be the first to be encountered during the search procedure. This may be accomplished by ordering the entries so that the probability density of the likelihood of finding a particular entry corresponding to a random text word is greatest at the points where the search is most likely to be proceeding.

This problem was considered by International Telemeter for the instance of a system using a single Photoscopic Disc. The problem will be considered in much more detail in the next few pages in order to demonstrate the relative merits of different ordering schemes in a more general system which uses two serial memories for the dictionary.

The proper relationship between the probability density of finding a random word in a particular region of the memory and the distance from the start of the search is easily determined. If the proability density is F(x), and x the distance from the start, then the probability of finding the word in an incremental distance, dx, is F(x), dx. The problem is then to arrange the word order in such a way that the first moment of the probability density function of the word order (F(x)) is minimized, or:

$$\bar{x} = \frac{\int_{0}^{\infty} x\ F\ (x)\ dx}{\int_{0}^{\infty} F(x)\ dx}$$

and since $^{\infty}_{0}\ F\ (x)\ dx = 1$ by the definition of probability density, $\bar{x}$ recuces to $\bar{x} = \int_{0}^{\infty} x\ F\ (x)\ dx$.

In order to minimize the average access time to information stored in the memory, a special alphabetical sequence must be devised to place the more common words near the beginning of the dictionary. To determine the desired alphabetical order the probability density of various letters must be obtained.

In Figure 6, the letters of the Russian alphabet which may start words are listed in standard alphabetical order. In the first column to the right of the letters is listed the number of words with this first letter in a 3671 text sample. For instance, 48 words starting with occurred, 377 with , 444 with , and so on. The second column lists the number of pages in Callaham's "Russian-English Technical and Chemical Dictionary," which are required to list the words starting with that particular letter. For instance requires 19 pages, and 13 pages. In order to minimize the first moment of the probability density function, a listing according to a probability density of words in the dictionary is required. This may be approximated by dividing column (1) by column (2) to give column (3), which has the dimensions of "occurrences in the text per page of dictionary." In Figure 6b the letters are reordered according to their value in column (3) of 6a. Note that the order is considerably different from standard alphabetical order. This ordering would be optimum for a single memory in which the search always started from the beginning. The calculations of Figure 7a show the benefit of ordering. Figure 7a shows an average search distance of 380 pages for ordering according to standard alphabetic sequence, while Fig. 7b shows that for optimum ordering only a 283 page search distance is indicated.

The calculations of Figs. 8a and 8b show the importance of search from the middle of a memory rather than from one end. In such a case the words of highest probability density must be placed near the most probable position of the search, or the middle of the memory. Figure 8a shows an example of double ordering i.e., the dictionary is stored in the individual memories in such a way that the probability of a random text word being located in either memory is very nearly .5. The order of Table IIIA places the letters of greatest probability density at the start of each memory. In contrast, the order of Table IIIB places the words of greatest probability density near the middle of each memory. The average search distance is considerably less in the second case; about 75 pages as contrasted to 139 for the case of search from the beginning of the memory.

Order is thus an important consideration in any non-random-access memory. In an actual case, optimum ordering might also depend upon the second or third letters of words as well as the first. It is not absolutely necessary that succeeding letters have the same code as the first: the input device could code in any desired manner. Dictionary search is only a matching problem so it is only necessary that the incoming text material be coded identically to the source language portion of the dictionary entries.

Table IA

Calculations of Search Distance for Various Types of Ordering (analysis of 3671 words)

	ALPHABETICAL ORDER				SINGLE ORDER	
	Occur.	Pages	Probability Density		Pages	Probability Density
а	46	26	1.77	е	2	26.5
б	90	32	2.81	и	13	14.7
в	377	52	7.25	ч	9	11.
г	48	36	1.33	н	36	7.63
д	179	32	5.6	в	52	7.25
е	51	2	26.5	о	49	6.04
ж	31	7	4.42	з	25	5.95
з	149	25	5.95	д	32	5.6
и	279	19	14.7	я	3	5.
к	215	54	3.97	т	37	4.85
л	27	19	1.42	о	75	4.6
м	137	38	3.6	э	13	4.38
н	275	36	7.63	ж	7	4.42
о	296	49	6.04	у	23	4.25
п	444	106	4.20	п	106	4.2
р	132	39	3.4	к	54	3.97
с	344	75	4.6	м	38	3.6

Table IA (continued)

т	180	37	4.85	р	39	3.4
у	98	23	4.25	ш	12	2.75
ф	25	16	1.56	б	32	2.81
х	25	12	2.08	х	12	2.08
ц	15	9	1.67	ю	1	2
ч	99	9	11.	а	26	1.77
ш	33	12	2.75	ц	9	1.67
щ	2	2	1	ф	16	1.56
э	57	13	4.38	л	19	1.42
ю	2	1	2	г	36	1.33
я	15	3	5	щ	2	1

Table IB

Occurrence of Words in Russian Text as a Function of the First Word

Double Ordering

е	26.5	и	14.7
ч	11.00	в	
н	7.63	з	5.95
о	6.04	д	5.6
я	5.00	с	4.6
т	4.85	к	3.97
э	4. 38	м	3.6
ж	4.42	ш	2.75
у	4.25	б	2.81
п	4.2	х	2.08
р	3.4	ю	2
а	1.77	ф	1.56
ц	1.67	щ	1.00
л	1.42		

Table IIA

Calculations
No Ordering, One Memory

	Ampl.	Width		Area	Arm	Moment
а	1.77	26		46	13	600
			58			
б	2.81	32		90	42	3780
			110			
в	7.25	52		377	84	31,700
			146			
г	1.33	36		48	128	6140
			178			
д	5.6	32		179	162	29,000
			180			
е	26. 5	2		51	179	9120
			187			
ж	4.42	7		31	183	5670
			212			
з	5.95	25		149	199.5	29,700
			231			
и	14.7	19		279	221.5	61,900
			285			
к	3.97	54		215	258	55,500
			304			
л	1.42	19		27	295	7950
			342			
м	3.6	38		137	323	44,200
			380			
н	7.63	36		275 164	360	99,000
о	6.04	49		296	402.5	119,000
п	4.20	106		444	480	213,000
			155			
р	3.4	39		132	552.5	73,000
			194			
с	4.6	75		344	609.5	210, 000
			269			
т	4.85	37		180	665.5	120,000
			306			
у	4.25	23		98	695.5	68,100
			329			
ф	1.56	16		25	715	17,900
			345			
х	2.08	12		25	729	18,200
			357			
ц	1.67	9		15	739.5	11,100
			366			
ч	11	9		99	748.5	74,100
			375			
ш	2. 75	12		33	759	25,100
			387			
щ	1	2		2	766	1,530
			389			
э	4.38	13		57	773.5	44,000
			402			
ю	2	1		2	781	1,562
			403			
я	5	3		15	782.5	11,730
					1392982 3671 =	
					380	

Table IIB

Single Ordered
One Memory

	Occ.		Width	Arm	Mom.
е	51		2	1	51
н	279		19	11.5	3210
ч	99		9	25.5	2520
и	275	66	36	48	13200
в	377	118	52	92	34700
о	296	167	49	142.5	42100
з	149	192	25	179.5	26800
д	179	224	32	208	37200
я	15	227	3	225.5	3400
т	180	264	37	245.5	44100
с	344	339	75	301.5	103800
э	57	352	13	345.5	19700
ж	31	359	7	355.5	11000
у	98	382	23	370.5	36300
п	444	488	106	435	193000
к	215	542	54	515	110500
м	137	580	38	561	77000
р	132	619	39	599.5	79100
ш	33	725	12	625	20600
б	90	757	32	741	66600
х	25	769	12	763	19100
ю	2	770	1	769.5	1539
а	46	796	26	783	36100
ц	15	805	9	800.5	12000
ф	25	821	16	813	20300
л	27	840	19	830.5	22400
щ	2		2	841	1682

$$\bar{x} = \frac{1038002}{3671} = 283$$

Table IIIA

Double Ordering with End Start

	Occ.		Width	Arm	Mom.
е	51		2	1	51
ч	99		9	6.5	742
н	275		36	29	7960
		47			
о	296		49	71.5	22200
		96			
я	15		3	97.5	1460
		99			
т	180		37	117.5	21300
		136			
э	57		13	142.5	8110
		149			
ж	31		7	152.5	4710
		156			
у	98		23	167.5	16400
		179			
п	444		106	232	103000
		285			
р	132		39	303.5	40100
		324			
а	46		26	337	15500
		350			
ц	15		9	354.5	5310
		359			
л	27		19	368.5	9960
и	279		19	9.5	2650
в	377		52	45	16950
		71			
з	149		25	83.5	12400
		96			
д	179		32	112	20000
		128			
с	344		75	165.5	56900
		203			
к	215		54	230	49400
		257			
м	137		38	276	37800
		295			
ш	33		12	301	9950
		307			
б	90		32	323	29100
		339			
х	25		12	345	8620
		351			
ю	2		1	351.5	703
		352			
ф	25		16	360	9000
		368			
щ	2		2	369	738

$$\bar{x} = \frac{511014}{3671} = 139$$

Table IIIB

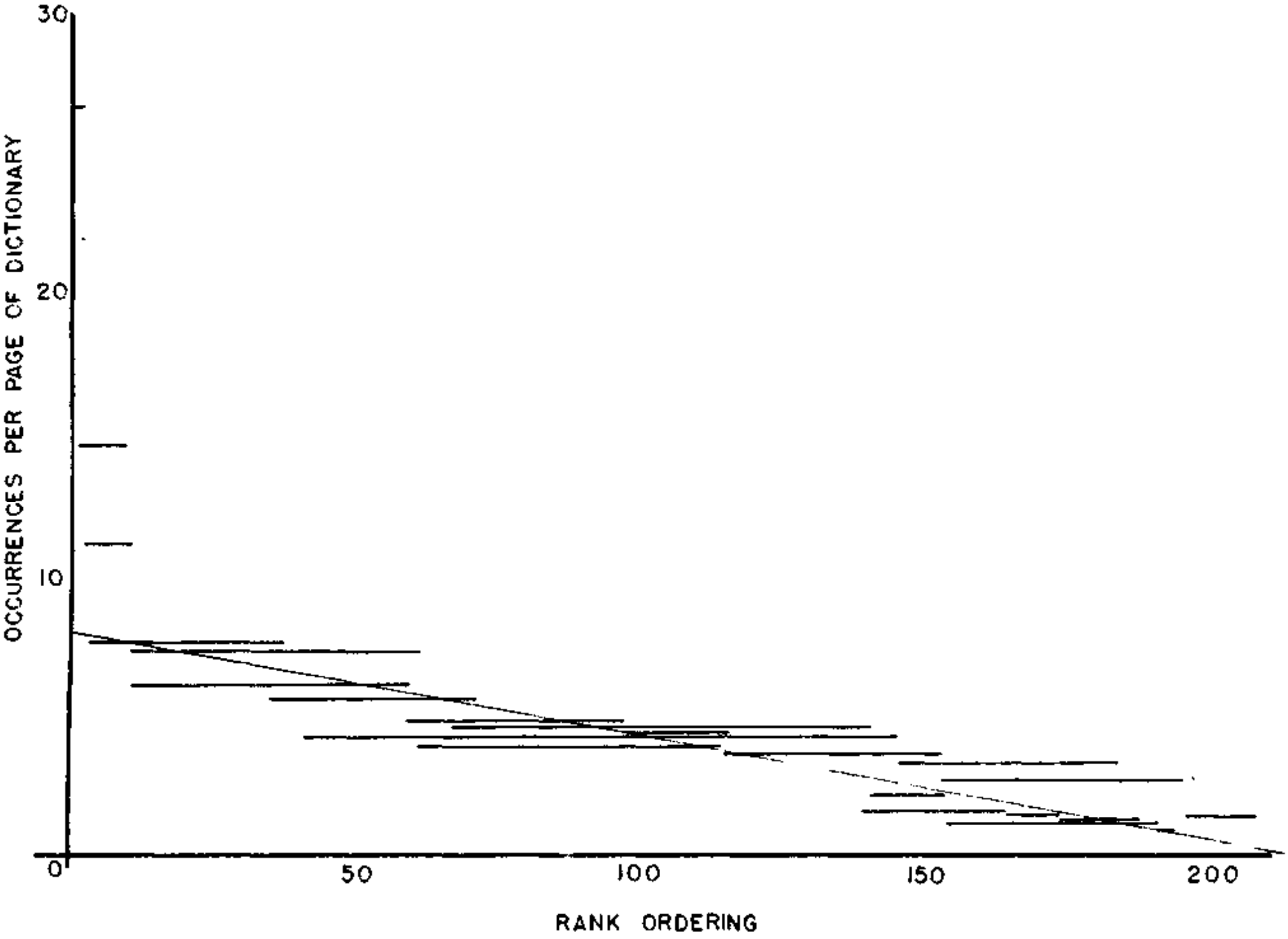
Double Ordering with Center Start

	Occ.		Width	Arm	Mom.
л	27		19	183.5	4950
		174			
ц	15		9	169.5	2540
		165			
а	46		26	152	7000
		139			
т	180		37	120.5	21700
		102			
у	96		23	90.5	8870
		79			
ж	31		7	75.5	2340
		72			
э	57		13	65.5	3730
		59			
о	296		49	34.5	10200
		10			
ч	99		9	4.5	445
		1			
е	51		2		0
		1			
н	275		36	19	5210
		37			
я	15		3	38.5	576
		40			
р	132		39	59.5	7850
		79			
п	444		106	132	58600
ф	25		16	205.5	5130
		197.5			
б	90		32	181.5	16320
		165.5			
ш	33		12	159.5	5260
		153.5			
м	137		38	134.5	18400
		115.5			
к	215		54	88.5	19100
		61.5			
в	377		52	35.5	13400
		9.5			
и	279		19		
		9.5			
з	149		25	22	3280
		34.5			
д	179		32	40.5	7250
		76.5			
с	344		75	114	39200
		151.5			
х	25		12	157.5	3940
		163.5			
ю	2		1	164	328
		164.5			
	25		16	172.5	4310
		180.5			
щ	2		2	181.5	363

$$\bar{x} = \frac{274297}{3671} = 75$$

PROBABILITY DENSITY OF
OCCURRENCE OF A WORD
IN A DOUBLE ORDERED
MEMORY WITH CENTER START

Figure 1



Appendix IV

Some of the Economics of Machine Translation

The costs which may be expected for machine translations are of considerable interest.

When the economics of a certain process are considered, it is necessary to compare the proposed method with other existing methods. In the case of MT, economic studies usually consist of comparisons between costs of material translated by machine and costs of human translation of the same material. These studies necessarily include a consideration of the advantages of both methods, since per-word cost of translation is not the only criterion for judging the feasibility of translation by machine.

MT has several advantages which for some applications might make it attractive even if it were considerably more expensive than human translation. For one thing, a much smaller staff would be required to perform large scale translations by machine, especially if an efficient electronic reader were available. One machine with an efficient electronic reader and two operators should be able to replace forty to sixty human translators. A staff of forty to sixty poses in itself a considerable personnel problem. In addition, the machine translator could be programmed to catalog automatically the material it translates according to key words or key sentences in the text. It is also true that the machine could perform translations extremely rapidly. All of these advantages might be more important than the cost difference if the application were, for example, in a field like military intelligence.

A comparison of the costs of human translation with estimated costs for machine translation is rather difficult. Machine translations contemplated at the present are considerably different from human translations, with the degree of difference varying with the type of material to be translated. The difference would be great in the case of mathematical texts including much demonstration of mathematical developments. It would thus be idle to compare costs of a polished human translation of such material with a hypothetical machine translation of the same material. Translators at the University of Washington who were consulted indicated that, for material which is not too highly specialized, the cost per word would probably be about one cent. It would seem, therefore, to compete on a strictly cost basis, a machine must be able to perform translations for not over one cent per word.

Several suggestions have been made for the design of a translation machine. In the elaboration of the design of such a machine one has to consider the translation process and the computer requirements for each step. There are four definite steps in the translation process:

1. Preparation of the text material for the machine.
2. Dictionary search and the production of the first crude word-for-word translation.
3. Logical processing to improve this first crude word-for-word translation.
4. Print-out of the improved word-for-word translation.

Two methods are available for preparation of the text material for introduction into a translator. The first is the use of typists copying the text material on punched or magnetic tape which is then fed into the machine. The second method requires the use of an electronic reader to read and code the text material automatically. A considerable effort has gone into the development of electronic readers in the last five years. Intelligent Machines Corporation and IBM have both demonstrated electronic readers, but no one has yet demonstrated an electronic reader capable of the high speed necessary for MT.

The experience of the UW research group would indicate that the preparation of material on tape by typists will cost between .75 and 1 cent per word. This estimate is based on a salary scale of $240 per month (a little below the standard for private industry in the Seattle area) and on the employment of competent typists who were graduate students in the Russian division of the Far Eastern Department of our University. Thus it is quite clear that an electronic reader must be developed if MT is to become commercially practicable.

An estimate of the cost of the various steps in the translation process is as follows:

1. Preparation of text material (by typists)	.85
2. Dictionary search (Photoscopic Memory)	.06
3. Logical processing	.50
4. Print out (Flexowriter)	.05
Total (cents per processed word)	1.46

Conclusions

The electronic reader is of tremendous importance for MT. Without such a device machine translation cannot be justified on the basis of cost alone. A really efficient electronic reader will reduce per-word costs below those of human translations. On the other hand, it is likely that, for many applications, MT will not have to be justified merely on a cost-per-word basis.

In the early stages of MT research most writers were concerned more about a memory with sufficient storage capacity than about any other component of a MT computer. It seems that with memories such as Telemeter Photoscopic Memory the problem of permanent storage is near solution. The main problem is now an electronic reader.