R E P O R T R E S U M E S

ED 012 014 AL 000 275

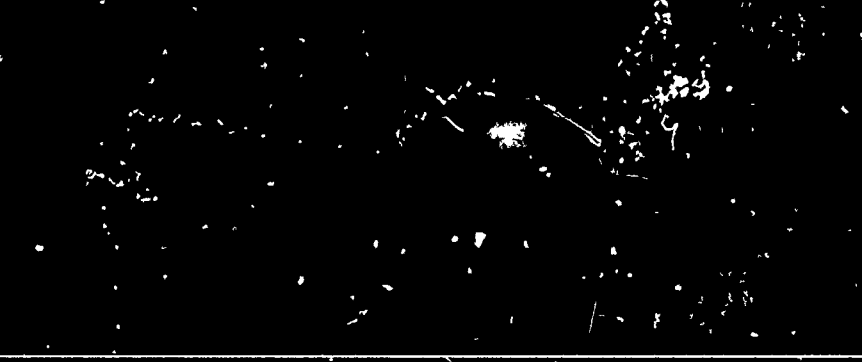
COLLECT--A PROGRAM FOR THE RETRIEVAL OF GRAMMATICAL INFORMATION FROM ANNOTATED TEXT.

BY- KAY, MARTIN TAFT, TERRIL D.
RAND CORP., SANTA MONICA, CALIF.

REPORT NUMBER RM-5243-RADC PUB DATE JAN 67
EDRS PRICE MF-$0.09 HC-$1.44 36P.

DESCRIPTORS- *COMPUTATIONAL LINGUISTICS, *RUSSIAN, *SYNTAX, *SEMANTICS, *INFORMATION RETRIEVAL, INFORMATION SYSTEMS, LINGUISTIC ANALYSES, GRAMMAR, IBM 7044 COMPUTER, THE COLLECT PROGRAM, SANTA MONICA

DESIGNED FOR LINGUISTS AND LEXICOGRAPHERS, THE COLLECT SYSTEM CAN ANALYZE THE SYNTAX AND SEMANTICS OF RUSSIAN SENTENCES BY SEARCHING A FILE OF RUSSIAN TEXT. IN A TYPICAL SEARCH ON A GIVEN SENTENCE, GRAMMATICAL RELATIONSHIPS, INDIVIDUAL WORD SPELLINGS, AND WORD OCCURRENCES CAN BE DETERMINED. SINCE THE SYSTEM INCLUDES INTERPRETERS, THE LINGUIST CAN USE FAIRLY NATURAL TERMINOLOGY IN HIS SEARCH REQUESTS. WRITTEN FOR THE IBM 7044 COMPUTER, THE ENTIRE COLLECT SYSTEM COULD BE ADAPTED TO ADMIT NEW KINDS OF INFORMATION TO THE FILE. THIS MEMORANDUM ALSO DISPLAYS A SAMPLE COLLECT REQUEST FORM. (FB)



MEMORANDUM
RM-5243-RADC
JANUARY 1967

COLLECT: A PROGRAM FOR THE RETRIEVAL OF GRAMMATICAL INFORMATION FROM ANNOTATED TEXT

Martin Kay and Terril D. Taft

This research is supported by the United States Air Force under Contract No. AF 30(602)-3812, monitored by Air Force Systems Command, Research and Technology Division, Rome Air Development Center, Griffiss Air Force Base, New York. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

The RAND Corporation
1700 MAIN ST. • SANTA MONICA • CALIFORNIA • 90406

PREFACE

A File of a million words of Russian text, annotated with syntactic information in the form of dependency structures, has been prepared by the RAND linguistics project for Rome Air Development Center. In preparation for syntactic (and also, ultimately, semantic) analyses of the material in this File, a system of computer programs called COLLECT has been written. Using this system, a linguist can write a description of a fairly complex phenomenon in fairly natural terms and obtain all, or a predetermined number of, instances of the phenomenon in the File.

Although the system was written for this specific File and the kinds of searches linguists can be expected to make, it is flexible enough to permit addition of new kinds of information to the File, enrichment of the request terminology allowed, and modification of report formats. It is also representative of a class of retrieval programs needed for many file-search tasks.

The present Memorandum is therefore of most interest to linguists and lexicographers who might use this File, but may also be of some interest to those designing and using other files.

SUMMARY

COLLECT is a system for searching a File of Russian sentences annotated with syntactic (and, eventually, morphological) information. The File consists of sentences; the user writes a description of the class of sentences he wishes to examine, and COLLECT edits his request, searches the File (taking several requests simultaneously, if desired), and prints a report. A sentence description consists of descriptions of word-form occurrences — their spellings, grammatical properties, etc. — and their relationships to one another. Small units of descriptions can be composed by conjunction or disjunction, and can be negated. The COLLECT system has been written for the IBM 7044 computer.

CONTENTS

COLLECT: A PROGRAM FOR THE RETRIEVAL OF GRAMMATICAL INFORMATION FROM ANNOTATED TEXT

1. INTRODUCTION

This Memorandum describes a computer program for collecting sentences with specified attributes from a large File of Russian text. In this File, each sentence is provided with a dependency structure; that is, grammatical relationships have been indicated, and the grammatical function each occurrence serves for its governor has been recorded. The structure of a whole sentence takes the form of a tree with a word at every node, an independent clause at the root.(1)

The File was prepared to serve the needs of linguists investigating the syntax and semantics of Russian. COLLECT permits the linguist to describe the kind of sentence that would provide evidence relevant to his current problem. The system includes interpreters, so that he can use fairly natural terms in describing his needs. It can save the results of a search; hence the linguist can proceed step by step from a broad description to a narrower one without having to pay for repeated searches of the complete File. The system is capable of marking selected word occurrences within the sentences it collects; the linguist can use the marked items as sort keys to arrange his material after collection. The system also includes a report generator.

The present version of the COLLECT system does not allow the linguist to refer to every significant kind of information that is, or could be, in the File; there are no sorting facilities; and only one report format is ready. The system has been designed, however, to allow for future modifications in these and other respects.

2. THE STRUCTURE OF A REQUEST

To use the COLLECT system, the linguist must write a request. In it he specifies what part of the File is to be searched (Sec. 4), how the material collected is to be dealt with (Sec. 5), and what attributes a sentence must possess to be of interest to him (Sec. 3).

In the File, a sentence is a sequence of occurrences. The properties of an occurrence are its form (the actual string of letters that occur at that place in the text), the punctuation marks before and after it, its grammatical properties as determined by consultation of a dictionary, and its relation to the rest of the sentence. The relational information attached to an occurrence consists of the occurrence number of its governor and the grammatical function that the present occurrence serves for its governor.

The linguist describes a sentence by stating properties of occurrences. He could request collection of all sentences containing nouns by specifying that in every sentence of interest there must be an occurrence with the

grammatical property of nounhood. COLLECT also permits the linguist to make use of relational properties; he can ask for each sentence that contains a noun depending on another noun by specifying two occurrences, each with the property of nounhood and one with the property of depending on the other.

COLLECT permits the linguist to write a complex request by combining simple statements. Each statement in isolation is taken as a propositional variable, either true or false when applied to a certain occurrence in a certain sentence. The linguist can indicate that several statements jointly describe a single occurrence; otherwise, each statement or marked group of statements must apply to a distinct occurrence. If several statements describe one occurrence, they can be disjunctive, so that an occurrence satisfies them collectively if it satisfies any of them, or conjunctive, so that an occurrence satisfies them collectively only if it satisfies each of them individually. In fact, these modes of combination can be used repeatedly; an individual statement can be negated; and one statement can specify several properties. For example, one statement can call for a noun followed by a comma and serving the function of grammatical subject. If that statement is negated, any occurrence satisfies it that is not followed by a comma; or, being followed by a comma, is not a noun; or being a noun followed by a comma, is not a grammatical subject; and so on.

The COLLECT request form (Fig. 1) provides space for a heading (labeled A in the figure), which must include a specification of the portion of the File to be searched. The description of what is wanted goes in the space labeled B; and the treatment of collected sentences is specified in space C.

3. ENCODING OF DESCRIPTIONS

A request to COLLECT is a description of a class of sentences. The linguist describes an abstract sentence, designating the form occurrences in it by letters: occurrence A must be a noun, B must depend on A, C must occur later than A in the sentence, and so on. In more complex situations, alternatives must be specified: occurrence A is a verb, B is either an adverb or a preposition. In writing requests, numbers are added to letters to indicate alternatives: occurrence A is a verb, B1 is an adverb, B2 is a preposition.

Each line of a COLLECT request is a statement describing a type of occurrence. In each statement, a symbol is given to identify that type. A symbol consists of a letter, optionally followed by a numeral. In testing a sentence, COLLECT attempts to find one occurrence for each letter; hence all statements bearing the same letter refer to a single occurrence in any sentence, and statements bearing distinct letters must be satisfied by distinct occurrences. If two statements bearing the same letter also have the same number, or no number, they must be satisfied jointly by one occurrence, but if the numbers

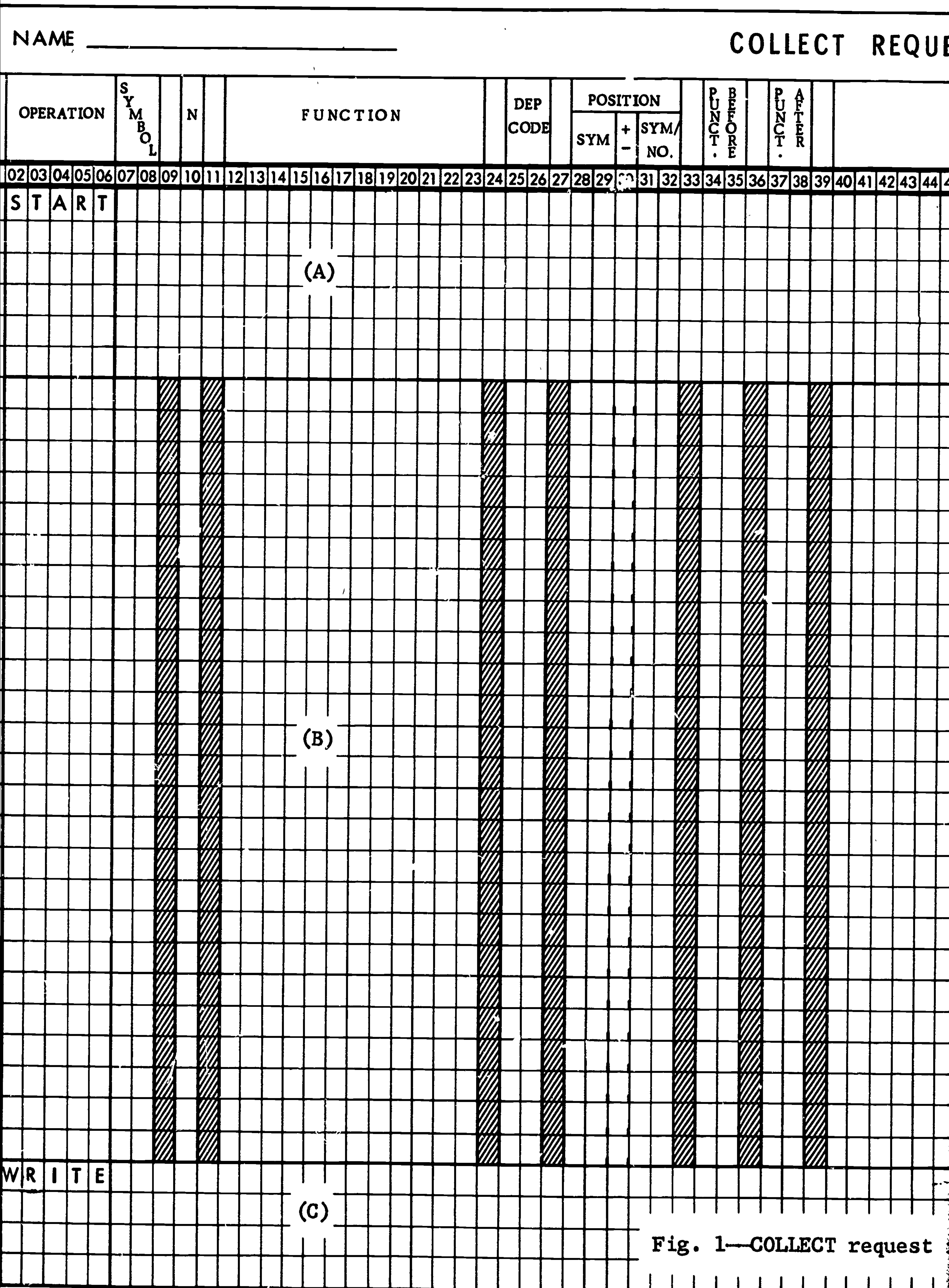


Fig. 1—COLLECT request

ECT REQUEST

AFTER		GRAMMAR CODE	IDENTIFICATION	SEQ. NO.
38	39	40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72	73 74 75 76 77 78	79 80
				0 0

LECT request form

are distinct, the statements are related disjunctively.

Consider the following statements:

A	(Property 1)
B1	(Property 2)
B1	(Property 3)
B2	(Property 4)
C	(Property 5)
C	(Property 6)

This request is satisfied by any sentence including three occurrences of which one bears property 1, a second bears properties 2 and 3, and a third bears properties 5 and 6; but it is also satisfied by any sentence including three occurrences, the first and third as before, the second bearing property 4.

The order of symbols in a request is not related in any way to the order of occurrences in a sentence, and a sentence satisfies a request if it includes occurrences satisfying the statements of the request, no matter how many more it includes. For example, a sentence would satisfy the request above if its 5th occurrence satisfied B, its 11th satisfied C, and its 24th satisfied A.

To specify that one occurrence must depend on another, the linguist writes the symbol of the governor in the dependency code column of the statement referring to the dependent (columns 25, 26 of the form; see Figs. 1 and 2). Thus, for example,

A		Verb
B	A	Noun

requires that some occurrence with the property of being

a noun must depend on another with the property of being a verb. If the dependent is described disjunctively, the symbol of the governor must be included on each line:

```
A        Verb
B1   A   Noun
B2   A   Pronoun
```

If the symbol A were omitted from the last line, any pronoun would satisfy the B requirement, whether governed by the verb called A or not.

A symbol must always be defined before it is used. If the first line of the last example were moved to the end, the request would be ill formed.

If the dependency code field is blank, the statement can be satisfied by an occurrence without regard to what it depends on. If a zero is written in this field, only a grammatically independent occurrence can satisfy the statement.

Figure 2 shows a request that will be answered by any sentence containing a noun that has among its dependents another noun in the genitive case. The first line calls for any noun; when COLLECT searches a sentence, it looks for a noun and upon finding one assigns its occurrence number as value to the symbol A. The second line calls for a noun in the genitive case, depending on the first noun found. The value assigned to A is used in seeking such a noun. If no occurrence meets the specifications, COLLECT looks for another noun, and, if it finds one,

changes the value of A and tries again to find a genitive dependent. In this way all possibilities are explored before the sentence is rejected.

In the function columns of the form (12-23 in the figures), the linguist can write any of the functions used in the File: M = modifier, S = subject, and so on.(1) A subroutine is used to match the formulation written here against the contents of the File; since the functional coding in the File is relatively complex, it seems that several different subroutines will be needed in the future — simple ones to be used when the linguist is able to describe what he wants in terms of relatively superficial properties, complicated ones to be used when essential. For example, if a verb has two conjoined nouns as subject, a special procedure is needed to discern the true relationship between each noun and the verb. Another kind of problem is raised by apposition; for some analyses, the linguist may choose to regard a noun in apposition to the subject of a sentence as serving subjective function. In a sentence like "Several plants, such as lichen and moss, grow in the Arctic", the occurrences of lichen and moss can be treated as serving subjective function, in some sense, even though that is not the grammatical function recorded for them in the File. Again, ellipsis occurs with some frequency, and for some purposes the linguist will prefer to analyze the structures reconstructed by the annotators.

As presently constituted, the File does not contain the kind of grammatical information obtainable from a dictionary. Until such information is added, the grammar–code field (columns 40–72 in the figures) is unusable. The example in Fig. 2 suggests some facilities that could reasonably be made available to the linguist. Subroutines, adapted to the encoding of whatever grammatical information is added to the File, and recognizing grammatical terminology convenient to the linguist, can be provided as necessary.

As we have remarked, the order of statements on the request form does not reflect the order of corresponding occurrences in text. From time to time, occurrence order may be important, and therefore it must be specifiable. The linguist uses the position field (columns 28–32 in the figures) for this purpose. He can require that one occurrence precede or follow another, or that one occurrence lie between two others, or within a fixed distance from the beginning or end of a sentence.

The first two columns of the position field (labeled SYM) can be used to fix on an occurrence, by writing in the symbol that stands for it. The next column is used to signal direction: after (+) or before (–) the designated occurrence. For example, suppose the linguist wants to find sentences in which verbs are modified by preceding

prepositional phrases; he can write

```
A                    Verb
B  M  A  A  -        Preposition
```

To qualify as being of type B, an occurrence must serve modifying function, depend on a type A occurrence, precede the type A occurrence, and qualify as a preposition. If the modifying phrase must precede the subject of the verb, the linguist can write

```
A                    Verb
B  S  A
C  M  A  B  -        Preposition
```

The third statement reads: An occurrence of type C serves function M, depends on an occurrence of type A, precedes (-) an occurrence of type B, and is a preposition.

The last two columns of the position field can contain nothing, a symbol, or a number. To specify that an occurrence must lie between two others, the linguist writes their two symbols; for example, if a modifier must lie between preceding subject and following verb, he writes

```
A                    Verb
B  S  A
C  M  A  B  A        Preposition
```

In this case, the $\pm$ column is left blank. A number in the last two columns of the position field sets an upper limit on the separation between two items. If the preposition must directly follow the verb, the linguist writes

```
A                          Verb
B  M  A    A + 1           Preposition
```

Any one or two digit number can be written in this space; if the number is n, there can be at most n - 1 intervening occurrences.

If no symbol is written in columns 28 and 29, but a plus or a minus sign in column 30 and a number in columns 31-32, the position is taken relative to the beginning or end of the sentence; thus + 1 refers to the first occurrence in a sentence, - 1 refers to the last.

Here, as in the dependency code field, a symbol can be used only if it has been defined earlier in the request.

The linguist can refer to punctuation by writing the marks of interest in columns 34-35 or 37-38. In the File, punctuation marks are recorded before an occurrence (e.g., open quotation mark, open parenthesis) or after (e.g., period, comma, close quotation mark, close parenthesis). In a request form, one or two marks of preceding punctuation and one or two marks of following punctuation can be written in the description of any occurrence: if two marks are written in the same field, the appearance of either in the File satisfies the request. Thus, a noun followed by both a period and) would be requested as follows:

```
A          .)          Noun
```

In writing punctuation marks, the linguist uses the appropriate Hollerith character according to the punctuation alphabet of the RAND text-encoding scheme.(2) A punctuation mark does not occupy an occurrence position in the File, hence does not count in determining the separation of two word occurrences, depends on nothing, can govern nothing.

The linguist can specify a word form in a request. He writes W in column 01, the flag column, and the word he wants in columns 7-72. On such a line he does not write a symbol, but instead lists the words just below the line containing the intended symbol. For example,

```
    A          Adjective
    B    A
W   PRI
W   PO
```

This request collects all sentences in which either of the word forms pri or po depends on an adjective. The linguist writes Russian forms in the Jakobson transliteration;(3) for other alphabets, he uses flags: *R for the Roman alphabet, *G for the Greek, *S for the symbol alphabet, and *C for Cyrillic. Except in the Cyrillic alphabet, he uses the Hollerith equivalents of the RAND encoding scheme.(2) Since an asterisk signals that the next character is an alphabet flag, a special representation of the asterisk itself is needed: **.

Several word forms can be written on a single line, separated by blank spaces. Several lines can be used for

a list, but a word form cannot be divided between two lines. The line preceding a list can specify properties in the ordinary way; an occurrence then satisfies the request only if it has the specified properties and matches one of the word forms listed.

Segments of word forms can also be specified. The wanted segments are listed in the same manner as full forms, but instead of the flag W in column 01, P is used for a prefix, S for a suffix, and I for an infix. Combining prefixes and full forms in one list (but not on one line), and using negation, the linguist could specify any form beginning with pri- except the full form prihela, for example.

Forms can be grouped in various ways, according to the stems on which they are based, according to meaning, and so on. When such a grouping is obtained from a dictionary, provision can be made to use group identification in collecting sentences. For example, all the forms of bytq constitute such a group: if a dictionary designates the group by number, then, after information from the dictionary is collated with that already in the File, a flag (to be used in column 01) can be defined, and linguists can be permitted to collect all occurrences of bytq by writing its designating number on a request line.

Any statement can be negated. To do so, the linguist writes N in column 10. If only one statement carries a

```
A
A  N      Noun
A  N      Verb
```

The first line is satisfied by every occurrence in the File, since it imposes no requirements whatsoever. The second is satisfied if the occurrence is not a noun, the third if it is not a verb. All three are satisfied by any occurrence that is neither a verb nor a noun, and a sentence is collected if it contains at least one occurrence that satisfies the request.

4. REQUEST HEADINGS

Above the description of what is wanted, the linguist writes a heading block. The first line always contains the word start, written in columns 02-06; this marks the beginning of a request, and is needed since several requests can be processed in one scanning of the File. From this line, the COLLECT system takes the identification provided by the user in columns 73-78; that identification appears in all references to this request, and all products that result from carrying it out.

After the start line, the user can write, in whatever terms he pleases, a description of what he is trying to do; this statement is copied in the report he gets. Columns 02-72 can be used for the purpose, but column 01 must contain either C or *. (Comments thus marked can be inserted anywhere in a request.)

The linguist must specify what portion of the File

is to be searched. He does so in the heading of his request beginning in column 07, using a line on which he writes in in columns 02–03. This statement can continue on subsequent lines, which must be blank in columns 01–06. If he wants to search the entire File, the linguist writes all. Otherwise, he writes a string of parameters, separated by commas. Each parameter consists of a corpus identifier, optionally followed (after a period) by a division label, and that optionally followed (again after a period) by a section label. For example, P.A.55 signifies the physics corpus, division A, section 55. Writing several parameters, e.g.

P.A.1, P.A.3, P.A.5

causes just the indicated sections to be searched. A hyphen can be used to request a search of several consecutive segments:

P.A–P.F

signifies that divisions A through F of the physics corpus are to be scanned. Before writing a request, the linguist must consult a list of corpora, divisions, and sections in the File.

5. FINAL INSTRUCTIONS

After describing the phenomenon he wants to study, the linguist adds some general remarks that control the form of the results he obtains. The main component of the COLLECT system scans the File, choosing sentences

that satisfy the linguist's description and writing them on a new tape. His final instructions determine how many sentences are written, whether counts are made, whether indicators are written to show how each sentence satisfies his description, and whether the output tape is saved after a report is printed for him. These instructions are written on a line marked by the word write in the operation field (columns 02-06).

The first instruction on the write line pertains to the indicators, or pointers. In his description, the linguist uses symbols — A, B, C, etc. to refer to distinct occurrences, and A1, A2, etc. to refer to a single occurrence that can satisfy his requirements with one set of properties or another. COLLECT will construct pointers, consisting of symbols and occurrence numbers, if the linguist writes down the symbols of interest. If he wants pointers for all symbols, he can request

WRITE ALL

and each sentence will be accompanied in the output by pointers to occurrences satisfying all his requirements. The pointers need not be listed in any particular order; COLLECT writes them on the output tape in the order given them by the linguist on the write line. Thus they can eventually serve as a key to control sorting: the first can specify what is to be a major sort, the next an intermediate sort, and so on. If the linguist makes no request, no pointers are created.

In general, a request can be satisfied by a single sentence in more than one way. COLLECT finds every possible way in each sentence, and writes a given sentence on the output tape with as many sets of pointers as are valid. For example, suppose the linguist is looking for any noun (A) with a genitive noun dependent (B), and that some sentence contains a string of four nouns with occurrence numbers 5, 6, 7, and 8, all genitive but the first. Then three sets of pointers are created: A–5, B–6; A–6, B–7; and A–7, B–8. If the linguist does not need all these pointers, he can specify <u>first</u>, and COLLECT will stop processing a sentence upon finding one way in which it satisfies the request.

The search of a million words of annotated text can yield a burdensome amount of output if the researcher underestimates the frequency of his phenomenon, or if the constraints he puts into his description are too loose. To avoid unwanted output, the linguist can write <u>maximum</u> (or abbreviate: <u>max</u>) and a number. No more than this number of sentences will be accepted by COLLECT; if the required number is reached before all of the text specified by the linguist has been scanned, the search terminates. If no maximum is specified in a request, COLLECT sets a maximum of 100; the linguist's maximum can be larger.

Occurrences of the phenomena of interest can be counted during the search. The linguist requests statistics (or stat) on the write line; only the occurrences satisfying symbols listed on this line are counted. If he writes statistics only, the output tape contains no examples of his phenomenon, merely counts. If first is specified, the counts are all identical to the number of sentences in which the phenomenon is detected, unless some requirements can be satisfied in different ways. Thus, if the description contains the symbols A, B1, B2, and C, the counts for A and C are sentence counts, and the sum of the frequencies of B1 and B2 is the same number; but in each sentence, either B1 or B2 is found first and tallied.

If the linguist wants to preserve a tape copy of the sentences collected, he specifies save on the write line. The pointers created will be retained on this tape only if the linguist specifies save with pointers. Otherwise the pointers appear on the linguist's printed report and then disappear.

A complete write line might include all or none of these specifications. Thus, for example,

```
WRITE     ALL, MAX = 200, SAVE
```

leads to a listing of 200 sentences (if there are that many in the text identified in the heading), each sentence accompanied by pointers to all occurrences satis-

fying parts of the request description, and an output tape with no pointers. Again,

WRITE FIRST

obtains up to 100 examples (automatic maximum), no pointers, and no permanent tape.

WRITE A B, STATISTICS, SAVE WITH POINTERS

causes a report to be printed with the frequencies of A and B (by subtypes, if any), pointers to occurrences satisfying A and B, and a tape with examples and pointers. But in this last example, only 100 sentences are collected.

6. SUMMARY OF REQUEST FORM

Requests to the COLLECT system are written on the form displayed in Fig. 1. The first line of a request must contain start in columns 02–06; this line, and others at the user's option, contains request identification in columns 73–78 and sequence numbers in columns 79–80.

A line with in in columns 02–03 is required. Here the user specifies what part of the File is to be searched.

A line with find in columns 02–05 marks the end of the heading: following lines are interpreted as description statements.

The end of a request is marked by a line with write in columns 02–06.

Comments can be introduced, on separate lines, anywhere in a request. A comment must have C or * in column 01.

7. FILE ORGANIZATION AND RETRIEVAL STRATEGY

COLLECT is intended for daily use on a File of a million words of annotated text. The system is really worthwhile only if several researchers can use it as often as necessary without having to be too concerned about its cost and without having to guarantee on each occasion that the information retrieved will be crucial or even useful. It must encourage invention by making exploration safe.

The File is recorded in catalog format;[4] the graphic form of each occurrence is written in the RAND text-encoding scheme. The latter is simply a method of representing the information from an ordinary printed page, with all its varieties of alphabet, font, and style of type, in a manner suitable for computer processing. A catalog is a file of information written on magnetic tape according to certain conventions that provide a hierarchical organization of the content. The catalog system offers the advantage of standardization across files, facilitating their use by researchers who were not directly involved in their creation and permitting the use of general programs not designed specifically for any one of them.

These advantages of the catalog system are outweighed, in the present context, by the urgency of making COLLECT an inexpensive system to run. COLLECT is, in any event,

specialized to the kinds of information in the File and the known requirements of linguists who will use it. Moreover, some of the information in the File can be set aside for the present. Leaving out some information and putting the rest in a compact format tailored to the operations COLLECT will perform can reduce by two-thirds the length of the File on tape. Since the amount of tape to be read is the most important factor in determining how long the program will run (and hence what it will cost to process each retrieval request), it seems appropriate to base COLLECT on a specialized format.

The first file written in COLLECT format contains text identification, the form of each occurrence, the primary dependency indicator and associated function code, and the negation indicator. Although the format is specialized, it is flexible enough to allow addition of all other information now in the basic File, information obtained by consulting a dictionary, and so on. The initial version permits test runs to be made early and allows timing trials for evaluation of the search algorithm.

The text identification consists of a two-character corpus identifier, a one-character division label, a numeric section identifier, and a page number, all in one word. The program that produces the file in COLLECT format writes a cross reference table for use by linguists, who must convert these abbreviated identifiers into those of the original File.

For simplicity and speed in comparing requested word forms with those of occurrences in the File, an encoding change is made. All font and type style shift characters are removed. No alphabet flags appear in the form of a Cyrillic occurrence; otherwise, the flag is the first character of the form. Punctuation is isolated and categorized as occurring before or after the form occurrence. Each occurrence is carried as an isolated entity. Other information is left unchanged.

The data about a sentence compose a logical record in the COLLECT format. These variable-length records are grouped into physical records of approximately the largest size permitted by the space available when the COLLECT program is run. The file is stored in ascending sequence by text identification.

Each request submitted to the COLLECT system is edited before the search. The editing routine checks the order of the start, in, find, and write sections and verifies that symbols used in the dependency code and position fields are defined prior to use. Several requests can be submitted for simultaneous processing; those passing editorial inspection are assembled in storage arrays that are passed on to the search program. Rejected requests are returned with diagnostic messages to aid the linguist in correcting them.

The search program reads the File sentence by sentence. It checks text identification to determine for each sentence whether it lies within the range of the request. It searches the sentence for the properties requested, and writes output if they are found.

The search algorithm works by backtracking. It starts with the first symbol mentioned in the request and finds the first occurrence in the sentence that has all the properties attributed to that symbol, recording that occurrence number as a value for the first symbol. The second symbol is taken next, and searching begins with the first occurrence in the sentence, but occurrences identified with symbols are excluded. If values are assigned to all symbols, the sentence satisfies the request in one way. When no value can be assigned to a symbol, or when a complete set of values has been assigned and noted, backtracking occurs. Suppose, for example, that a request includes symbols A, B, and C in that order. Suppose, further, that A-2, B-5, and C-7 is an acceptable assignment. After this result is written out, if the sentence consists of more than seven occurrences, scanning for later occurrences that can satisfy the C specification continues. If none is found, the next step is to continue the scan for a type B occurrence and, upon finding another, to restart the scan for a type C occurrence at the beginning of the sentence. Each acceptable assignment results

in output and is followed by further scanning, until every possible assignment has been found.

The output format is identical with that of input, except that the pointers can be added. Statistics are accumulated during the search and can be written out at the end.

One output tape is generated during the search, no matter how many requests are being processed. After the search, the tape is sorted by request identification, and processed by a report writer that produces hard copy and saves results as requested. The current program can only print out complete sentences, pointers, and counts. Other formats, and additional sorts prior to listing, can be added as experience dictates.

REFERENCES

1. Hays, D. G., "Dependency Theory: A Formulation and Some Observations," Languages, Vol. 40, No. 4, October - December 1964, pp. 511-525; also The RAND Corporation, RM-4087-PR, July 1964.

2. Kay, M., and Ziehe, T. W., Natural Language in Computer Form, The RAND Corporation, RM-4390-PR, February 1965.

3. Jacobson, R., "O latinizacii mehzdunarodnyx telegramm na russkom hazyke," Voprosy Hazykoznaniha, Vol. 14, No. 1, 1965, pp. 111-113.

4. Kay, M., Valadez, F., and Ziehe, T. W., The Catalog Input/Output System, The RAND Corporation, RM-4540-PR, March 1966.