AUTOMATED ANALYSIS OF LANGUAGE STYLE AND STRUCTURE IN TECHNICAL AND OTHER DOCUMENTS

Technical Report #1

Sally Y. Sedelow, Principal Investigator

September, 1971

This research was supported by
the Office of Naval Research

through

Contract N00014-70-A-0357-0001
under Project No. NR348-005

University of Kansas
Departments of Computer Science and Linguistics
Lawrence, Kansas 66044

AUTOMATED ANALYSIS OF LANGUAGE STYLE AND STRUCTURE IN TECHNICAL AND OTHER DOCUMENTS

Sally Y. Sedelow, Principal Investigator
H. William Buttelmann, Consultant
Martin Dillon, Consultant
Walter Sedelow, Consultant
Frank Joyce
Thomas Kosakowski
Peggy Lewis
David Wagner
Sam Warfel
Harrel Wright

TABLE OF CONTENTS

AUTOMATED ANALYSIS OF LANGUAGE STYLE AND STRUCTURE IN TECHNICAL AND OTHER DOCUMENTS

Sally Yeates Sedelow, Principal Investigator

ABSTRACT

The project concerned with Automated Analysis of Language Style and Structure in Technical and other Documents has had four primary emphases during the 1970-71 research year. One emphasis has been the development of approaches to the modeling of thesauri, so that existing thesauri can be precisely described and new thesauri (more adequate to a wide range of information retrieval requirements) can be developed. A second emphasis has been the writing of statistical data-gathering programs and the testing of statistical analytical tools on that data; this work is directed toward a statistical support package for natural language research. A third emphasis has been upon beginning the development of a theory of prefixing in English in order to provide a basis for automated handling of prefixes. The fourth emphasis has been upon programming a FORTRAN IV, Honeywell 635 version of the list-structure VIA programs, which are used for content analysis as well as for inputs to the research on thesauri and statistical analysis.

PREFACE

I would like to thank the Information Systems staff at the Office of Naval Research and the ONR Resident Representative at the University of Kansas for their helpfulness concerning the administration of this contract. I am also grateful to the Office of Research Administration, The Computer Science and Linguistics departments, and the Computation Center at the University of Kansas for their assistance with various aspects of the contract. And I am especially indebted to the Computer Science department secretaries for their fine efforts throughout the year and most especially during the preparation of this report.

I. Survey of Project in Automated Analysis of Language Style and Structure in Technical and Other Documents September 1, 1970 - August 31, 1971

A. Introduction

The Automated Analysis of Language Style and Structure project is directed toward the development of methodology and models for computer-based analysis and, ultimately, generation of natural language. The explosion of information and man's inability to cope with its mass is evermore abundantly apparent. Much of this information is encoded in natural language; but man's knowledge of the structures produced by the coding is still not sufficient to enable him to construct good, comprehensive models for language analysis or even relatively adequate partial models. One thrust of this research project is to explore language structure and style through a wide range of dimensions (e.g., lexicographic distributions, semantic clusterings, positional distributions of morphemes, etc.) in an effort to approach more comprehensive models of language--models which would more adequately facilitate information analysis in its myriad variety, including the special needs of the intelligence information analyst.

Models which we seek would thus provide a basis for assumptions concerning normative language usage by an entire culture or subculture, as well as allowing for individual deviations from the cultural norm. Thus, although there is a need in the case of many applications (e.g.,

automatic abstracting, document retrieval, subject matter retrieval) for a more precise description of the normative use of, for example, English, there is also a need to be able to discriminate between and among users of English. The latter form of discrimination can tell us when a document written by one hand has been tampered with by another or others, thus perhaps indicating a shift in point of view, or emphasis.

Many computer-based language analysis or generation projects have been ad hoc for a particular application, or for a subpart of a larger application area (e.g., chemistry for information retrieval), and thus have lacked the power of generalization to other applications or a larger application area. On the other hand, many professional linguists now tend to emphasize theory based on a few, isolated examples and are hostile to the extensive testing and empiricism made possible by the computer. This Automated Analysis of Language project emphasizes both general-purpose methodology and the extensive empirical testing of the methodology through the use of the computer, which, in fact, makes the methodology possible. This kind of approach is, we believe, widely vital if we are to achieve satisfactory methods for dealing with information and the language in which it is encoded.

B. Summary of Past Year's Work

Efforts this past year have been directed toward continuing work on 1. the structure and content of thesauri; 2. the development of a package of statistical programs for language analysis; 3. a theory of prefixing; 4. rewriting and extension of existing programs in the VIA package.

The research on thesauri has evolved from a continuing effort to automate fully the VIA programs, which are used for content analysis, thematic analysis, or any investigation of groups of semantically-related words in any text, document, transcribed oral discourse, etc. Because VIA is concerned with semantic relationships, it has been necessary for the user of the package to prepare lists (for one version) or, sometimes, categories (for one aspect of the ring-structure version) of semantically-related words for submission to the final program of the VIA package--the program which searches the text and prints out lists or rings of semantically-related words which occur in the text. The prepared lists were based upon thesauri, synonym dictionaries, context, and any other relevant references or user experience. In order to obviate the necessity (but not the possibility) of preparing such lists, research upon thesauri and synonym dictionaries was initiated. That research (described in detail in S. Y. Sedelow, et al, Stylistic Analysis, Third Annual Report, March 1, 1967, DDC # AD 651-591; S. Y. Sedelow, et al, Automated Language Analysis, Report on research for the period March 1, 1967 to February 29, 1968, DDC # AD 666-587; S. Y. Sedelow and W. A. Sedelow, "Categories and Procedures for Content Analysis in the Humanities," in The Analysis of Communication Content, ed. Gerbner, et al, John Wiley & Sons, Inc., 1969, pp. 487-499) indicated the desirability of getting dictionaries and thesauri into computer-accessible form. The thesaurus which appeared most suitable for extended investigation was Roget's International Thesaurus, which was keypunched; computer accessible copies of Webster's Collegiate Dictionary, Pocket Dictionary, and the Random House Dictionary were also obtained. The

simple mechanics of getting these reference works into usable form has, itself, been extremely time-consuming. Roget's Thesaurus can now be used experimentally but it cannot be exhaustively analyzed because error proofing is still in progress. The Webster's Dictionary tapes will soon be usable at the University of Kansas, but the reliability of the Random House tapes (which we also have at Kansas) is questionable. The need for techniques and formal approaches to the description of such reference works early became apparent--if the structure of a work is not understood it is impossible to identify reliably its biases and thus to modify the work so that it is more suitable for whatever task is being undertaken. Thus, while this research project was still based at the University of North Carolina at Chapel Hill, David Wagner undertook a doctoral dissertation which would, in part, attempt to deal with the parametric nature of thesauri. Professor Martin Dillon joined the project as a consultant and, upon the move of the Principal Investigator to Kansas, became officially associated with the thesaurus research as it has been carried on at Chapel Hill. In "Models of Thesauri and Their Applications," they report on the progress of that aspect of the research. As their article makes clear, this research is of major importance to any work connected with information retrieval, and, in general, with computer-based language analysis and generation. At present, there are no good, general-purpose stores of semantic information (e.g., thesauri) which can be used to produce both theoretically- and precisely-understood results for any computer-based language analysis or generation. This phase of the research is directed toward meeting that need.

Stylistic analysis--a major concern of the Automated Analysis of Language Style and Structure project--is often considered to be heavily

dependent upon statistical analyses of the document or text being studied. Such dependence has been less critical for this project, because, although the VIA programs do indeed receive some impetus from frequency counts, their emphasis is upon semantic relationships among words in any given text. Nonetheless, it is quite clear that statistical analyses are invaluable aids to the recognition and discrimination of stylistic patterns; for several years, this project has been working on the development of a package of statistical programs which will both gather and analyze natural language data. A first thrust in this direction was a cooperative effort with the research group under the direction of Walter Sedelow (sponsored by NASA) to inventory the many ad hoc approaches to statistical analysis of given texts. From this inventory, grew an effort to work out a taxonomy of approaches to data gathering and analysis (described by Martin Dillon in S. Y. Sedelow, et al, Automated Analysis of Language Style and Structure, Report on research for the period March 1, 1969 to August 31, 1970, DDC # AD 711-643). In "Toward a Statistical Support Package" and "Statistical Analysis of Linguistic Frequency Data" Frank Joyce (graduate student, Computer Science, K.U.) and Harrel Wright (graduate student, Psychology, K.U.) respectively, discuss their work on specific types of data gathering and analysis.

In a discussion of "Recognizing Roots in Words Which Have Prefixes" (Report on research for the period March 1, 1968 to February 28, 1969), S. Y. Sedelow pointed out a number of lacunae in the theory and knowledge of prefix behavior which posed hazards for the users of computer-based programs for prefix recognition. Sam

Warfel (graduate student, Linguistics, K.U.) describes (pp. 76-96) the initial phases of his effort to develop models for prefixes, which would then form a basis for systematic approaches to computer-based prefix recognition.

Owing to its intrinsic usefulness for content analysis and because it is an important tool for research on thesauri, prefixing, and certain linguistic statistics (since it provides a number of data categories, such as roots, affixes, semantic categories, etc.), it was important to make the VIA package available to the research group at Kansas. Since the PL/I version would not operate on the Honeywell/GE 635 (which has no PL/I compiler), the programs have been rewritten in FORTRAN IV in a highly modular fashion, so as to facilitate their use at other installations. The section on Program Documentation (pp. 97-184) contains Tom Kosakowski's (graduate student, Computer Science, K.U.) documentation of the FORTRAN IV version of INDEX, Frank Joyce's write ups of the PREFIX and SUFFIX programs, and Peggy Lewis' (graduate student, Computer Science, K.U.) outline of the list structure THESAUR program. The latter is still undergoing modification. The others are operational and program listings are provided in the appendix. One option now available in the list structure VIA which was not included in earlier versions is the capability of saving the output from THESAUR, so that the computer can be used to compare output from, for example, different chapters of a book or different sections of a document.

C. Plans for Future

During this next year, the work on formal approaches to thesauri and analogous structures will continue. Proofreading of Roget's International Thesaurus will also continue so that reliable statistics can be gathered and descriptions of the structure of the Thesaurus provided.

The effort directed toward developing a statistical package for natural language analysis will involve further design and programming of data gathering routines and their integration with other programs upon which they are dependent for data. On the theoretical side, statistical analytic models will be tested on data samples in the continuing search for models which seem best suited for natural language data. A special category of analysis which has been of particular interest for some time to the Automated Language Analysis Project is that of positional analysis (re any selected elements in a language string), and we expect to pay considerable attention to problems and possible solutions in this area.

Research toward a general model or theory of prefixes will also continue. Prefixing is important (and troublesome) to computer-based language analysis because prefixes carry a very heavy load of semantic information (e.g., anti, dis, con). To group together words having a common root, prefixes must be recognized; but on the other hand, the nature of the semantic information borne by any given prefix must also be recognized. At present, both the task of recognition and the task of interpreting what has been recognized are ad hoc, ill-explicated procedures--where they exist at all.

In addition to the statistical programs previously mentioned and any utility programs necessary for the research outlined above, this year will see the check-out of the FORTRAN IV list-structure version of the VIA program as well as continuing work on the ring-structure VIA. The preliminary sections of the ring-structure VIA have already been recoded in FORTRAN IV. Rather than redoing the remainder of this version for batch-processing, we mean instead to develop an interactive version which will enable us to experiment with different search procedures and different output formats. A major difficulty with the batch version was the enormous quantity of output. Using an interactive version, we hope to be able to watch various types of searches in progress so as to select modes which provide more manageable and thus more informative output.

D. 1. MODELS OF THESAURI AND THEIR APPLICATIONS

by Martin Dillon and David J. Wagner

1. Background

1.1 Types of Thesauri and Their Use

1.1.1 Language Specific Thesauri (Roget's)

Traditionally, a thesaurus is a collection of words arranged into groups of synonyms or near-synonyms. The organizing principles are semantic, depending on concepts of ideas rather than the form of words or occurrence patterns in the language. In order to show the relationship of concept groups to one another, the structural apparatus of the thesaurus is often extended by cross-referencing devices, generally dominated by the "see also," which is used to relate a group, or one of its members, to another group. Roget's International Thesaurus (Thomas Y. Crowell Co., New York, 1962), considered here as typical of this type of thesaurus, contains an elaborately structured index for cross-referencing purposes, and it imposes on its entries and categories a hierarchical structure which further extends its semantic organization.

With its origins in the nineteenth century, and intended primarily as a stylistic aid for the writer, Roget's presumably gathers together words in the language which are judged to be of value for contemporary use, and reflects in its structuring, as far as it can, actual structural linkages in the language. Beyond the practical value to its users, there seems little to guide an assessment of the principles of design or detail of such a device, either in general or for some specific example; unlike dictionaries, they have received little attention from linguistics,

---

The research reported here was supported in part by funds made available through the Space Sciences Program of the University of North Carolina. We are indebted to Mrs. Jane R. Harriss for her intelligent and energetic assistance in the preparation of this report.

primarily, one may suppose, because until only recently the semantics of natural language has been so little studied.

The application for such thesauri which predominates here is as a support in the automated description of natural language text. To improve this application, both in its conceptual and theoretical undergirding and in the development of more effective procedures and techniques, it is highly desirable to extend or invent models which treat such entities, either linguistically or computationally, with formal rigor. The concepts to be developed below are guided by these aims but can be considered independently of them: they focus on the task of clarifying the thesaurus as a complex data structure.

1.1.2 Discourse Specific Thesauri for Use in Information Retrieval

Thesauri as used in information retrieval have tended to take a somewhat different path because of the role they play in the retrieval process. They are two-edged: they are used to control the indexing of documents, where standardization is highly desirable for the sake of consistency, and for retrieval, where a potential user must be guided in his use of terms painlessly. In fully automated systems, lack of consistency is not the problem. Rather, there exists a need to offset consistently imprecise indexing by more flexible management of the interface with the user, where maneuvering within the conceptual structure is required.

Since thesauri provide implicitly a semantic structure for the document space they index, they are helpful in proportion to their potential for expressing structure. Such structure can be syntactic--where terms are considered to reflect a category depending on the

syntax of their individual occurrences--or semantic, where terms are presented within a structure capable of discriminating relations of greater complexity than mere synonymy. Examples of both kinds are described in Gerard Salton's Automatic Information Organization and Retrieval (McGraw-Hill Book Company, New York, 1968), where categories are defined through terms qualified by some syntactic relationship or where terms are treated as members of a hierarchy, and instances of terms in the lower branches imply the use also of higher branches. In the latter, such structure is normally used to control the level of a search, expanding the number of documents retrieved by moving up the hierarchy, and contracting them by moving down the hierarchy.

A more ambitious effort (and too complex to be dealt with here) is the SYNTOL experiments (J. C. Gardin, Syntol (Rutgers, 1965)), where the basic rules which determine category membership allow much more variety, most notably in the form of the relationship among terms.

1.2 Formal Models of Thesauri

An abstract thesaurus is a formal system

$$\underline{T} = \langle \underline{C}, \underline{W}, \underline{t}, \underline{d}, \underline{e} \rangle$$

where

$\underline{C}$ is a non-empty finite set

$\underline{W}$ is a non-empty finite set

$\underline{t}$ is a partial map $\underline{t}$: $\underline{C} \times \underline{C} \rightarrow \underline{Z}$

($\underline{Z}$ = the non-negative integers)

$\underline{d}$ is a partial map $\underline{d}$: $\underline{W} \times \underline{W} \rightarrow \underline{Z}$

and $\underline{t}$ and $\underline{d}$ are partial metrics; that is,

i) $\underline{t}\ (c_1, c_2) = 0$ iff $c_1 = c_2$

$\underline{d}\ (w_1, w_2) = 0$ iff $w_1 = w_2$

(reflexivity)

ii) $\underline{t}\ (c_1, c_2) \geq 0$ if $\underline{t}$ is defined for (c_1, c_2)

$\underline{d}\ (w_1, w_2) \geq 0$ if $\underline{d}$ is defined for (w_1, w_2)

(non-negativity)

iii) $\underline{t}\ (c_1, c_2) = \underline{t}\ (c_2, c_1)$

$\underline{d}\ (w_1, w_2) = \underline{d}\ (w_2, w_1)$

(symmetry)

iv) $\underline{t}\ (c_1, c_2) + \underline{t}\ (c_2, c_3) \geq \underline{t}\ (c_1, c_3)$

for all c_1, c_2, c_3, where $\underline{t}$ is defined

$\underline{d}\ (w_1, w_2) + \underline{d}\ (w_2, w_3) > \underline{d}\ (w_1, w_3)$

for all w_1, w_2, w_3, where $\underline{d}$ is defined

(transitivity)

and $\underline{e}$ is a relation: $\underline{e} \subseteq \underline{W} \times \underline{C}$

Intuitively, an abstract thesaurus $\underline{T}$ may realize a concrete thesaurus by interpreting $\underline{C}$ as the set of all categories, $\underline{W}$ as the set of all tokens, $\underline{t}$ as a (partial) metric between categories, $\underline{d}$ as a (partial) metric between words, and $\underline{e}$ as the relation "belongs to" listing the categories of which a word is a member.

Moreover, a concrete thesaurus may have several abstract thesauri which realize it. In particular, Roget's International Thesaurus is realized by the standard interpretation and also by defining:

$\underline{C}$ as the set of qualified hierarchy names and

$\underline{W}$ as the set of thesaural categories

and, in fact, one can define a sequence of abstract thesauri which

realize a single concrete thesaurus by defining

$$\underline{T}_i = \langle \underline{C}_i, \underline{W}_i, \underline{t}_i, \underline{d}_i, \underline{e}_i \rangle$$

where

$\underline{T}_0$ is the standard interpretation and

$\underline{T}_i = \langle \underline{C}_i, \underline{W}_i, \underline{t}_i, \underline{d}_i, \underline{e}_i \rangle$

$\underline{W}_i = \underline{C}_{i-1}$ and

$\underline{C}_i$ is a subset of the power set of $\underline{W}_i$ $(\underline{C}_i \subseteq \underline{P}\ (\underline{W}_i))$

In the particular case of Roget's, six different abstract thesauri can be immediately identified and directly interpreted.

Although $\underline{t}$, $\underline{d}$ and $\underline{e}$ admit many meaningful definitions, the following is representative:

i) $\underline{e}\ (w,c)$ is true if and only if word w belongs to category c

ii) $\underline{d}\ (w_1, w_2) = 1$ if $(\exists c)\ (\underline{e}(w_1, c)\ \&\ \underline{e}(w_2, c)\ \&\ w_1 \neq w_2)$

$\underline{d}\ (w_1, w_2) = \min_w \{\underline{d}(w_1, w) + \underline{d}(w, w_2)\}$

iii) $\underline{t}\ (c_1, c_2) = 0$ if $c_1 = c_2$

$\underline{t}\ (c_1, c_2) = 1 + \min_{w_1, w_2} \{\underline{d}(w_1, w_2) \mid \underline{e}(w_1, c_1)\ \&\ \underline{e}(w_2, c_2)\}$

1.3 Goal of Formal Analysis of Thesauri

The thesaurus seems an intuitively appealing model for meaning in natural language, and for this reason we expect it to assume a growing importance in language research. While this model competes to some extent with that represented by dictionaries (given more formal treatment by Katz and Fodor in "The Structure of a Semantic Theory" (Language, Vol. 3, no. 2, 1963, pp. 170-210)), and thus thesauri compete with dictionaries in practical applications, neither is sufficient by itself. Moreover, document retrieval studies and

stylistic analyses have come to depend on thesauri far more than on dictionaries, and the reasons for this are worth investigating.

The key seems to be the interdependence of terms and the structures such interdependence leads to. In stylistic analysis, it is not only the text which is at issue, but the use in the text of certain of the available structures in a language at the expense of others. In document retrieval, it is not merely the meaning of a specific term which determines what should be retrieved, but also how that term interacts with others throughout a set of texts. Thesauri reflect this interaction admirably, with the additional (short term) advantage that they do so in a manner simple enough to exploit through the use of computers.

Assuming such a point has been made, there are many reasons for requiring a more developed set of tools for working with these structures. Only three will be considered here: 1) to modify existing thesauri; 2) to compare two thesauri; 3) and to guide the development of new thesauri.

1) Modifications to existing thesauri are necessary for a number of reasons, though primarily their intent is usually to merge, condense, or disambiguate groups of terms. These operations help to improve the quality of a thesaurus for either stylistic or retrieval studies. They lend themselves to the goal of optimizing the use of a thesaurus, or determining the relative usefulness of competing thesauri for some specific task. Formal definitions of thesaural structures aid in designing such operations, and in assessing their outcome.

2) Since thesauri have not yet received sufficiently detailed formal treatment, there are few ways through which such structures can be compared, either independently of some specific application, or in the effectiveness with which they perform some task. Thus one goal for improving the means of explicitly describing thesauri is to provide a means for comparing and contrasting them. The approaches outlined below are geared toward this end, and include models with quantifiable counterparts where such comparisons may be made more easily.

3) Little progress has been made in devising clear and effective procedures for constructing thesauri, either with human skills, computers, or some combination of the two. A paucity of serious attempts may be the primary reason for this lack. In any event, providing descriptive measures of thesaural structure or devising ways of relating thesauri to dictionaries or coherent discourse is partly intended to shed light on the obscure area of construction of thesauri.

3. Selected Approaches to Thesaural Description

Introductory Remarks

The following sections present approaches to thesaural descriptions accompanied by tests or examples as far as was practicable with materials in hand. They are divided into three sections, corresponding to: 1. models which stress the thesaurus by itself, concentrating on internal structure and relationships, and the strength of such relationships; 2. models which relate thesaurus to thesaurus, where

the goal is to develop comparative measures, or operations which take as arguments two or more thesauri; 3. models which relate thesauri to other complex structures, primarily dictionaries and the language or samples of the language from which the thesaurus has been derived.

For clarification and to give some indication of the value of the ideas discussed, where convenience permitted, the measures and operations are exemplified by concrete examples. Three bodies of structured data are used for the purpose: Roget's Thesaurus, three dictionaries (Webster's Seventh New Collegiate Dictionary, (G. C. Merriam Co., Springfield, Mass., 1970), The Random House Dictionary of the English Language (Random House, New York, 1969), The American Heritage Dictionary of the English Language (Houghton Mifflin, New York, 1968)), and selected judicial opinions of the Supreme Court. The first three are used because they are available in machine-readable form and represent important general instances of the structures in question. The last is used as the example for text because the universe of discourse represents a highly structured use of language, where a premium is placed on exactness of expression.

3.2.1 Basic Measures of Words and Categories

A thesaurus is composed of words and categories, and the first relations of interest are between words, words and categories, and from category to category. Three different approaches will be discussed: the first two lead to measures which are primarily descriptive, and give information about the relative size of categories, their content, and the distribution of weights over the structural

hierarchy; the third attempts to provide connectivity measures between words or between categories.

Category Membership

The most basic descriptive measures for thesauri depend only on the concept of category membership, and equivalence classes defined on the size of a category, or the number of categories each word belongs to. Figure 1* depicts a frequency distribution derived from the categories of Roget's Thesaurus.

Measures of entropy or information content, derivable from such distributions, are also of interest for the words or for the categories. The entropy of a word is associated with uncertainty over which category is active for an instance of its use. The entropy of a thesaurus can then be defined as the average entropy of its words. The amount of expected information from an instance of use of word in a thesaurus is a function of the entropy of the thesaurus: if there are few words and many categories, the relative information is low. This interpretation views the use of words as selectors of thesaural categories, restricting attention to the relationship of words to categories.

Weighted Tree

A second descriptive model useful for thesauri like Roget's which define a hierarchy for its categories is to devise a weighting scheme over the nodes of the hierarchy. Roget's has a maximum depth of six in its conceptual arrangement of terms, as represented in figure 2. To derive a weighted tree, one defines a function over categories which associates a numerical quantity with each entry, and a procedure

*Figures for this article appear on pages 27-47.

for combining such values as one moves higher in the hierarchy. The result of applying such a procedure is represented in figure 3, where a weight of one unit is defined for each term and through a process of summation the weight of each category, and then each higher level is determined. Other possibilities include differentiating classes of words by syntactic function for example, or deriving weights from the frequency of use of the terms.

The purpose of such a representation is to provide a condensed view of the distribution of terms or weights over the substantive areas defined by the thesaural hierarchy. Such a weighted tree provides insight into the emphasis of the thesaurus with respect to labels of the tree. In the case of Roget's, which purports to reflect a language, the weighted tree represents the facilities of the language as defined by the organization of the thesaurus.

Word and Category Connectivity

Connectivity measures between words or categories are useful for many reasons. They help guide category merges, division, or elimination by showing where such actions least upset the existing structure. Though optimal distribution of words among categories seems dependent solely on the universe of discourse from which the thesaurus is derived, recent research suggests reasons why this may not be so. (We are thinking here of Malcolm Rigby's experience with classifying terms in information systems with rapidly expanding document collections, as reported in "U. D. C. in Mechanized Subject Information-Retrieval," Subject Retrieval in the Seventies-New Direction (to be published)). Below, such issues are reduced to considering definitions and measures of ambiguity.

It is possible to associate an ambiguity measure with words and categories of a thesaurus in the following way. For words which occur in more than one category, ambiguity, with respect to the thesaurus, depends on the relationship of the categories to which they belong. For categories which are close to one another (intuitively, let us say, but finally as measured by one of the techniques to be described), the ambiguity is of a low order; for categories a great deal different from one another, the ambiguity is of a high order. A possible measure, thus, is one which reflects this distance between categories.

A path between two categories C(i) and C(j) is defined to be a linkage from some word in C(i) → C(a) → to C(j). The length of the path is defined to be the number of intervening categories (n) + 1. The strength of the association between C(i) and C(j) is now defined to be a function of the average length of these paths, normalized both for words in C(i) which are not ambiguous (belong to no other category, and thus cannot relate to C(j)), and with respect to the length of each path. This latter is necessary so as not to distort the measure by one or more entries which link to C(j) but do so over a large number of categories.

2. A second way to disambiguate depends solely on categories and the length of the shortest path between them. This concept depends on the universe of path lengths in a specific thesaurus, where some range, normalized say to 0-1.0, is interpreted as a direct measure of ambiguity. Thus, if the actual path lengths range from 0 to N, ambiguity is considered to be scaled in the same range.

3. A third and perhaps more fruitful way is depicted in figure 4. (We are indebted to Dr. Stephen Weiss for suggesting this approach to us.) The rows are formed from categories which intersect at least by the word appearing in the upper left corner; the columns are formed from the union of the categories, with column numbers referring to the words listed beside the table. An 'X' in a row-column intersection means that the column word appears in the category whose name is given to the row. Given such a table, a variety of clustering techniques might be tried for automatically discriminating the groupings which are apparent to the eye. For the table headed 'POWER", for example, the word 'potency', registered in column 5, appears in 5 categories.

3.2.2 Thesaurus to Thesaurus Operations and Measures

Like most of the relations dealt with here, those between thesauri have not been defined with sufficient precision. There is no clear way to describe relationships among thesauri since they are not usually derived from the same universe of discourse. Even in those rare cases where two thesauri have been developed for the same subject field, the procedures and rules which define categories and their terms are usually too obscure to derive formal relationships from them.

There are two ways in which this difficulty may be overcome. First, if the procedures for constructing thesauri are given more explicit formulation, they may provide a basis for deriving meaningful relationships. Second, if procedures are developed which

relate thesauri to some other complex structure, either a dictionary or body of text, as discussed in 3.2.3, then such procedures themselves may be such a basis. Salton, for example, reports attempts to derive thesauri automatically in his Organization.

Lacking such explicit bases, mappings from one thesaurus to another can be carried out through comparative analyses of category content. Given measures of category affinity similar to those developed above, thesauri may be compared by deriving measures between their respective categories. These possibilities will not be further explored here because of the lack of computational experience in their application.

3.2.3 Thesaurus to Other Complex Structures

Introductory Remarks

By complex structure we mean such entities as dictionaries, bodies of text presumed to contain coherent discourse, or corpora representative of a language or universe of discourse. The purpose of developing procedures which move between such structures and thesauri should be obvious: they would provide a way of validating existing thesauri, or generating new ones.

One more limited, immediate goal is to provide more internal structure for categories and to relate categories to one another in more complex ways. In most thesauri, the words within one category are presumed to be of similar meaning, defined rather vaguely. More precise would probably be the designation that such words are strongly related, or associated with one another. In

retrieval work, the implication is that the words are equivalent with respect to a set of documents.

Since it is not likely that thesauri will extend their power without more detailed relationships than these, it is imperative to develop them in a setting where they gain precision. The linear measures developed above in 3.2.1 require extension by measures based on more complex structures than are yet available.

Dictionaries

As a start, and ignoring the problem of thesaurus generation, we may ask how the words in a category are related to one another from the perspective of some complex structure, such as a dictionary. Are the definitions of each of the words in a category related? How are they related? Is it possible to extend the linear structure of a category by introducing a set of syntactic relationships which tie the category more closely together? If a definition chain is followed, originating from some word as root, extracting the definition of each word in the definition, will the words in a category be linked together?

Formal procedures can be developed to construct answers to most of the above questions. Figures 5 and 6, carried out by hand exemplify this process. Figure 5, the easier to achieve, records the depth of the association, where the number in a row-column intersection represents the number of definition expansions, starting with the word in the row, which were required to reach the word in the column. Where no number is recorded, a linkage was not found after 5 levels. Given such a matrix for each category, a measure, perhaps

the average depth of linkage over the words of a category, could be derived to reflect the integrity or coherence of the categories with respect to some dictionary. For the thesaurus, there would be an average coherence over all its categories, leading to a means for comparing thesauri with respect to some dictionary.

Figure 6 represents the nature of the link in addition to its depth. The purpose here is to refine the concepts of linkage, association, or category membership, both to aid in constructing thesauri and to understand better how to apply them.

Specific Texts

The weighted semantic tree derived in 3.2.1 above to represent the thesaural hierarchy can be extended to represent textual material, either from language use generally, or, as it is used here, selected text from a restricted universe of discourse like judicial opinions. The use of Roget's to reflect the semantic structure of a set of judicial opinions appears more like an application of the thesaurus than a clarification of thesaurus structure, although under special circumstances, the combination could be used for this purpose--to investigate coverage in Roget's of the legal vocabulary, for example. There are two ways of viewing this application: as a means of representing the text with respect to the thesaurus, or as a means of representing the thesaurus with respect to the text. Which perspective should be highlighted depends on what is under investigation.

The technique is useful only for thesauri with hierarchical structure, such as Roget's, and the purpose is to reduce the intersection of the text and thesaurus to a weighted tree structure similar to that used in 3.2.1. The difference lies in the origin of the

weights for each category, where in this case they are a function both of the thesaurus and the text. The simplest, selected here for the sake of an example, is to use the frequency of occurrence in the text of the entry as the weight, summing over a category to arrive at its weight.

Figure 7 is a tree structure derived from such a procedure as applied to selected judicial opinions of the Supreme Court, showing only that portion of the tree equivalent to the segment in Figure 3. The configuration of the tree and the relative weights of the nodes reflect the use of language facilities in these opinions from the perspective of Roget's Thesaurus.

As with most such measures, their interest is primarily comparative, with the absolute values of the weights less significant. Studies are underway which analyze the evolution of such structures over time and among different Justices.

5. Concluding Remarks

The efforts described here can be extended in obvious ways, and need extension if they are to be fruitful to thesaurus development and use. The most important developments are computational: some of the procedures described above must be programmed and carried out by computer over far larger data bases. The results of applying these models to thesauri, dictionaries, and texts must be investigated and interpreted. There is a long way to go.

FIGURE 1

Frequency Distribution:
Categories by Number of Words in Category

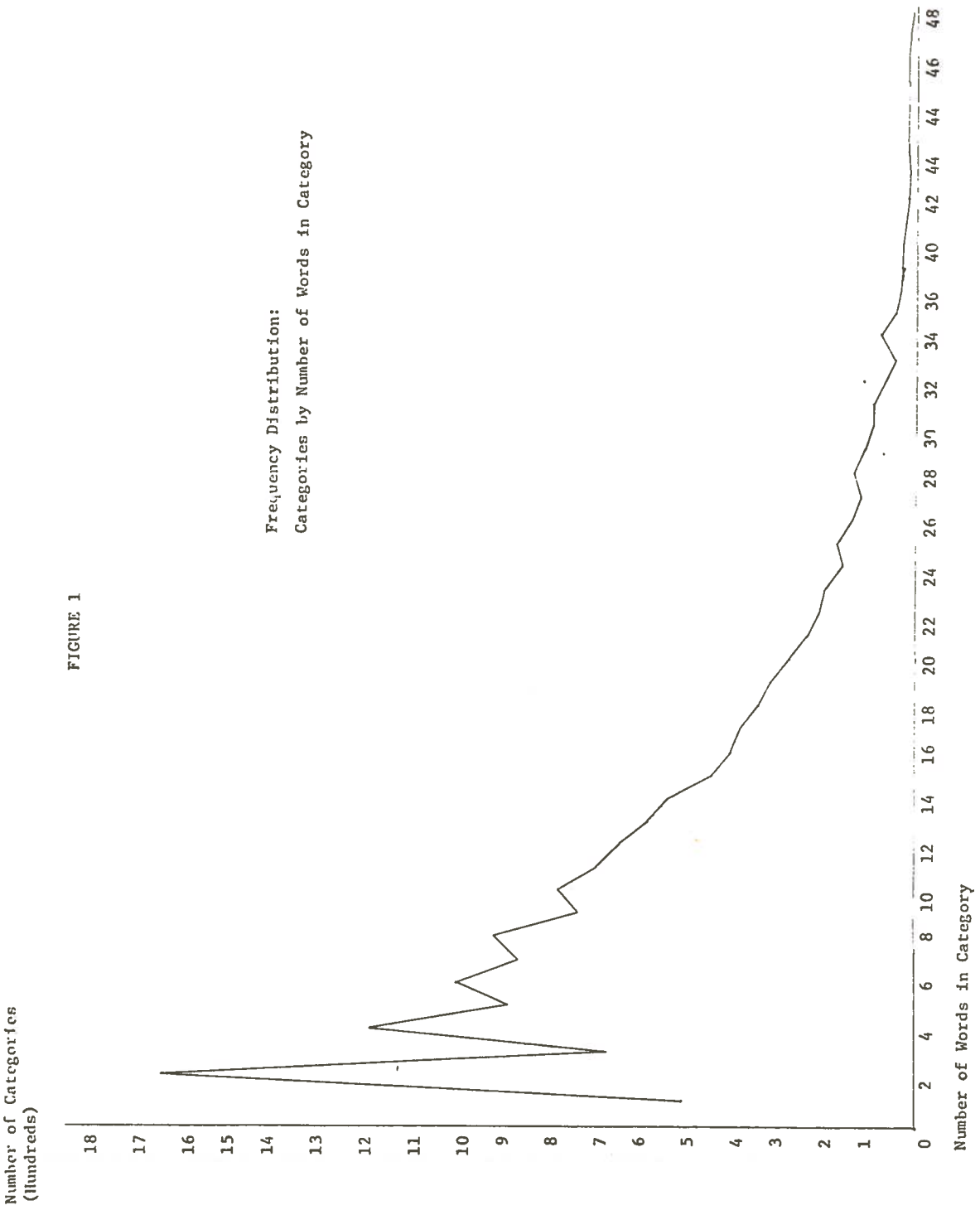
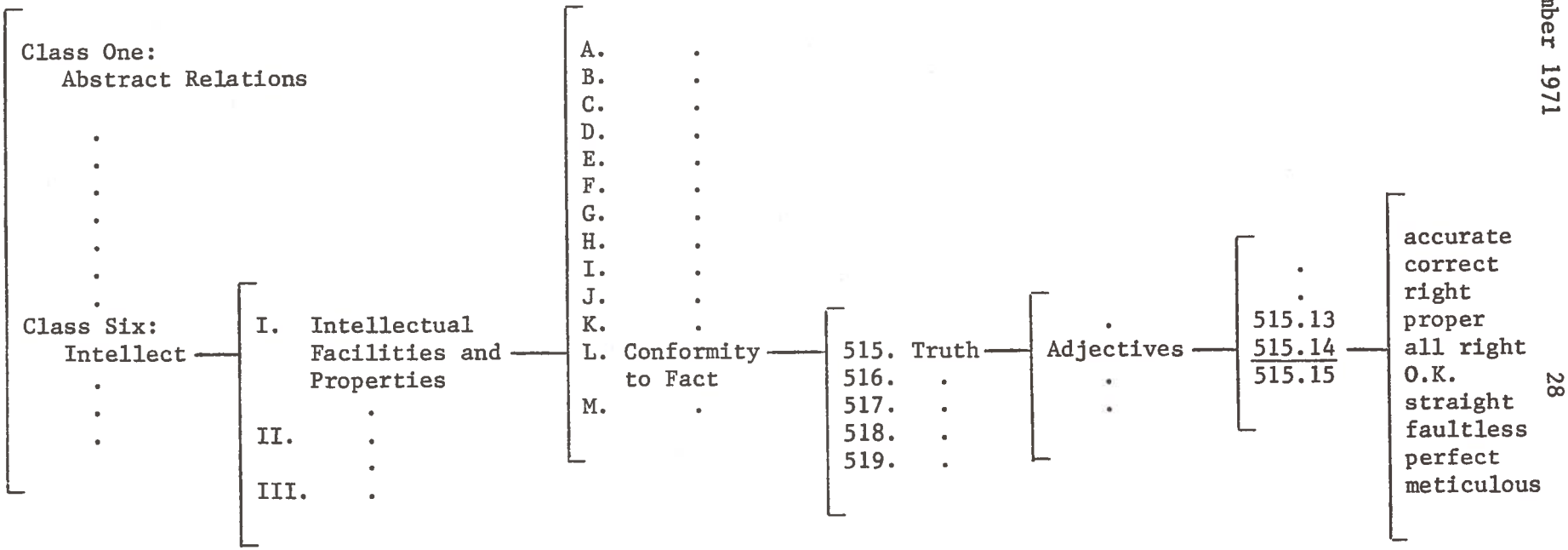


FIGURE 2



The location of subcategory 515.14 is shown, thereby demonstrating the hierarchial structure of Roget's Thesaurus.

FIGURE 3 (a)

CLASS SIX:INTELLECT

I.INTELLECTUAL FACULTIES AND P

A.FACULTIES

465	INTELLECT	89
466	INTELLIGENCE,WI	334
467	WISE MAN	64
468	UNINTELLIGENCE	363
469	FOOLISHNESS	273
470	FOOL	177
471	SANITY	57
472	INSANITY,MANIA	603
473	ECCENTRICITY	70
*****	SUM OF LEVEL 3	2030

B.COMPREHENSION

474	KNOWLEDGE	364
475	INTELLECTUAL	68
476	IGNORANCE	294
*****	SUM OF LEVEL 3	726

C.FUNCTIONS OF THE MIND

477	THOUGHT	317
478	IDEA	84
479	ABSENCE OF THOU	41
480	INTUITION	80
*****	SUM OF LEVEL 3	522

D.REASONING PROCESSES

481	REASONING	365
482	SOPHISTRY	229
*****	SUM OF LEVEL 3	594

E.CONSIDERATIONS

483	TOPIC	93
484	INQUIRY	668
485	ANSWER	94
486	SOLUTION	89
487	DISCOVERY	145
*****	SUM OF LEVEL 3	1089

F.ASSESSMENT

488	EXPERIMENT	166
489	MEASUREMENT	194
490	COMPARISON	136
491	DISCRIMINATION	134
492	INDISCRIMINATIO	67
*****	SUM OF LEVEL 3	697

G.CONCLUSION

493	JUDGMENT	209	
494	PREJUDGMENT	47	
495	MISJUDGMENT	24	
496	OVERESTIMATION	14	
497	UNDERESTIMATION	35	
*****	SUM OF LEVEL 3	329	
H.THEORY			
498	THEORY,SUPPOSIT	227	
499	PHILOSOPHY	130	
*****	SUM IF LEVEL 3	357	
I.BELIEF			
500	BELIEF	427	
501	CREDULITY	104	
502	UNBELIEF	229	
503	INCREDULITY	56	
*****	SUM OF LEVEL 3	816	
J.GROUNDS FOR BELIEF			
504	EVIDENCE,PROOF	388	
505	DISPROOF	165	
*****	SUM OF LEVEL 3	553	
K.QUALIFICATIONS			
506	QUALIFICATION	251	
507	NO QUALIFICATIO	49	
508	POSSIBILITY	159	
509	IMPOSSIBILITY	122	
510	PROBABILITY	101	
511	IMPROBABILITY	33	
512	CERTAINTY	412	
513	UNCERTAINTY	405	
514	GAMBLE	337	
*****	SUM OF LEVEL 3	1869	
L.CONFORMITY TO FACT			
515	TRUTH	416	
516	MAXIM	108	
517	ERROR	287	
518	ILLUSION	112	
519	DISILLUSIONMENT	61	
*****	SUM OF LEVEL 3	984	
M.ACCEPTANCE			
520	ASSENT	496	
521	AFFIRMATION	200	
522	NEGATION,DENIAL	147	
*****	SUM OF LEVEL 3	843	
*****	SUM OF LEVEL 2	11409	6.55%

II. STATES OF MIND			
A. MENTAL ATTITUDES			
523	MENTAL ATTITUDE	104	
524	BROAD-MINDEDNES	169	
525	NARROWMINDEDNES	155	
526	CURIOSITY	82	
527	INCURIOSITY	31	
528	ATTENTION	368	
529	INATTENTION	134	
530	DISTRACTION	293	
531	CARE	296	
532	NEGLECT	323	
*****	SUM OF LEVEL 3	1964	
B. CREATIVE THOUGHT			
533	IMAGINATION	373	
534	UNIMAGINATIVENE	76	
*****	SUM OF LEVEL 3	449	
C. RECOLLECTION			
535	MEMORY	336	
536	FORGETFULNESS	122	
*****	SUM OF LEVEL 3	458	
D. ANTICIPATION			
537	EXPECTATION	163	
538	INEXPECTATION	172	
539	DISAPPOINTMENT	82	
540	FORESIGHT	95	
541	PREDICTION	196	
542	FOREBODING	285	
*****	SUM OF LEVEL 3	993	
*****	SUM OF LEVEL 2	3864	2.22%
III. COMMUNICATION OF IDEAS			
A. NATURE OF IDEAS COMMUNICATED			
543	MEANING	165	
544	LATENCY	135	
545	MEANINGLESSNESS	230	
546	INTELLIGIBILITY	186	
547	UNINTELLIGIBILI	270	
548	AMBIGUITY	34	
549	FIGURE OF SPEEC	83	
550	INTERPRETATION	259	
551	MISINTERPRETATI	64	
*****	SUM OF LEVEL 3	1426	
B. MODES OF COMMUNICATION			
552	COMMUNICATION	150	
553	MANIFESTATION	260	

554	DISCLOSURE	196
555	INFORMATION	274
556	NEWS	184
557	PUBLICATION	315
558	COMMUNICATIONS	238
559	MESSENGER	67
*****	SUM OF LEVEL 3	1684

C.EDUCATION

560	TEACHING	278
561	MISTEACHING	41
562	LEARNING	210
563	TEACHER	132
564	STUDENT	139
565	SCHOOL	192
*****	SUM OF LEVEL 3	992

D.INDICATION

566	INDICATION	497
567	INSIGNIA	169
568	RECORD	293
569	RECORDER	23
*****	SUM OF LEVEL 3	982

E.REPRESENTATION

570	REPRESENTATION	182
571	MISREPRESENTATI	52
*****	SUM OF LEVEL 3	234

F.ARTS OF DESIGN

572	ART	344
573	SCULPTURE	99
574	CERAMICS	60
575	PHOTOGRAPHY	195
576	ENGRAVING	249
577	ARTIST	126
*****	SUM OF LEVEL 3	1073

G.LANGUAGE

578	LANGUAGE	272
579	LETTER	159
580	WORD	139
581	NOMENCLATURE	180
582	ANONYMITY	30
583	PHRASE	64
*****	SUM OF LEVEL 3	844

H.GRAMMAR

584	GRAMMAR	154
585	UNGRAMMATICALNE	37

***** SUM OF LEVEL 3		191
I.STYLE:MODE OF EXPRESSION		
586	DICTION	41
587	ELEGANCE	116
588	INELEGANCE	68
589	PLAIN SPEECH	55
590	CONCISENESS	76
591	DIFFUSENESS	145
***** SUM OF LEVEL 3		501
J.SPOKEN LANGUAGE		
592	SPEECH	315
593	IMPERFECT SPEEC	129
594	TALKATIVENESS	212
595	CONVERSATION	164
596	SOLILOQUY	25
597	ELOCUTION,PUBLI	158
598	ELOQUENCE	138
599	GRANDILOQUENCE	191
***** SUM OF LEVEL 3		1332
K.WRITTEN LANGUAGE		
600	WRITING	383
601	PRINTING	213
602	CORRESPONDENCE	148
603	BOOK,PERIODICAL	413
604	TREATISE	97
605	COMPENDIUM	76
***** SUM OF LEVEL 3		1330
L.LINGUISTIC REPRESENTATION		
606	DESCRIPTION	290
607	POETRY	441
608	PROSE	53
609	DRAMA	747
610	ACTOR	162
***** SUM OF LEVEL 3		1693
M.UNCOMMUNICATIVENESS:SECRECY		
611	UNCOMMUNICATIVE	117
612	SECRECY	304
613	CONCEALMENT	234
***** SUM OF LEVEL 3		655
N.FALSEHOOD		
614	FALSENESS	560
615	EXAGGERATION	118
616	DECEPTION	447
617	DECEIVER	194

	618	DUPE	70	
	*****	SUM OF LEVEL 3	1359	
*****		SUM OF LEVEL 2	14296	8.21%

TOTAL IN CLASS SIX:INTELLECT 29569 16.99%

FIGURE 3 (b)

CLASS SIX:INTELLECT

I.INTELLECTUAL FACULTIES AND P		
A.FACULTIES	2030	
B.COMPREHENSION	726	
C.FUNCTIONS OF THE MIND	522	
D.REASONING PROCESSES	594	
E.CONSIDERATIONS	1089	
F.ASSESSMENT	697	
G.CONCLUSION	329	
H.THEORY	357	
I.BELIEF	816	
J.GROUNDS FOR BELIEF	553	
K.QUALIFICATIONS	1869	
L.CONFORMITY TO FACT	984	
M.ACCEPTANCE	843	
***** SUM OF LEVEL 2	11409	6.55%
II.STATES OF MIND		
A.MENTAL ATTITUDES	1964	
B.CREATIVE THOUGHT	449	
C.RECOLLECTION	458	
D.ANTICIPATION	993	
***** SUM OF LEVEL 2	3864	2.22%
III.COMMUNICATION OF IDEAS		
A.NATURE OF IDEAS COMMUNICATED	1426	
B.MODES OF COMMUNICATION	1684	
C.EDUCATION	992	
D.INDICATION	982	
E.REPRESENTATION	234	

F.	ARTS OF DESIGN	1073	
G.	LANGUAGE	844	
H.	GRAMMAR	191	
I.	STYLE:MODE OF EXPRESSION	501	
J.	SPOKEN LANGUAGE	1332	
K.	WRITTEN LANGUAGE	1330	
L.	LINGUISTIC REPRESENTATION	1693	
M.	UNCOMMUNICATIVENESS:SECRECY	655	
N.	FALSEHOOD	1359	
*****	SUM OF LEVEL 2	14296	8.21%

TOTAL IN CLASS SIX:INTELLECT 29569 16.99%

FIGURE 4

Category Connectivity

(1)

CONTROL	1	2	3	4	5	6	7	8	9	10	11	12
162.1	X	X			X							
758.1		X	X	X	X							
162.6	X	X			X							
758.7		X	X	X	X	X	X					
171.1								X	X			
737.4								X	X		X	X
739.2									X	X	X	X
171.8						X		X				
488.5							X					

1 = moderation
2 = constraint
3 = inhibition
4 = curb
5 = restraint
6 = govern
7 = check
8 = rule
9 = power
10 = hold
11 = jurisdiction
12 = command

(2)

1 = interest
2 = title
3 = claim
4 = vested interest
5 = prerogative
6 = power
7 = authority
8 = faculty
9 = due
10 = fit
11 = correct
12 = proper
13 = decorous
14 = good

RIGHT	1	2	3	4	5	6	7	8	9	10	11	12	13	14
808.4	X	X	X	X										
956.3	X	X	X	X	X	X	X	X	X					
737.1					X	X	X	X						
974.8									X	X		X		X
515.14											X	X	X	
956.8									X	X	X	X	X	X
643.7											X	X	X	

(3)

POWER	1	2	3	4	5	6	7	8	9	10	11	12	13	14
156.1	X	X	X	X	X	X								
158	X	X	X	X	X									
160	X	X	X		X	X								
598.3	X	X			X	X								
956.3							X	X	X	X				
739.2											X	X	X	X
737.1							X	X	X	X				
171.1					X	X				X	X	X	X	X

1 = vigor
2 = strength
3 = energy
4 = might
5 = potency
6 = force
7 = faculty
8 = right
9 = prerogative
10 = authority
11 = control
12 = mastery
13 = domination
14 = hold

---

(4)

1 = pressure
2 = prestige
3 = weight
4 = moment
5 = consequence
6 = eminence
7 = domination
8 = predominate
9 = sway
10 = bias
11 = move
12 = induce
13 = persuade
14 = prompt
15 = incline

INFLUENCE	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
737.3	X	X	X	X	X	X									
171.1	X	X	X	X	X	X	X	X	X						
171.11							X								
739.14							X	X							
171.7									X	X	X	X	X	X	X
525.8									X	X					
646.22-23									X	X	X	X	X	X	X

FIGURE 5

1. Webster's Collegiate

	1	2	3	4	5	6	7
1	X	1	2	3			
2	1	X	1	2			
3	2	1	X	1			
4	1	1	1	X	3	2	
5					X		
6	1	2	3	4	1	X	
7							X

1 = accurate
2 = correct
3 = right
4 = proper
5 = faultless
6 = perfect
7 = meticulous

The words used are taken from category 515.14 of Roget's. The title over each table refers to the dictionary used in determining the depth of the association.

2. Random House

	1	2	3	4	5	6	7
1	X	1	2	2	2	2	1
2	1	X	1	1	1	1	2
3	1	1	X	1	2	2	2
4	1	1	2	X	2	2	2
5	3	2	3	3	X	1	4
6	2	1	2	2	2	X	3
7	3	2	3	3	3	3	X

3. American Heritage

	1	2	3	4	5	6	7
1	X	1		2			5
2	1	X		1			4
3	2	1	X	1			5
4	2	1		X			5
5					X		
6	1	2		2		X	2
7							X

FIGURE 6

The natures of the linkages found in the Random House table (Figure 5, (2)) are given here. The types of linkages are:

a = synonym
b^1 = modified by adverb
b^2 = modified by prepositional phrase
b^3 = modified by adjective
c = a property of some object

	LEVEL 1	LEVEL 2	LEVEL 3	LEVEL 4
1. accurate →	meticulous (a) correct (c) →	right (c) perfect (a) proper (a) faultless (a)		
2. correct →	right (c) proper (a) faultless (a) perfect (a) accurate (a) →	meticulous (a)		
3. right →	good (b^2) proper (b^2) accurate (a) →	meticulous (a)		
4. proper →	accurate (a) → correct (a) →	meticulous (a) right (b^3) faultless (a) perfect (a)		
5. faultless →	perfect (a) →	correct (b^2) →	right (c) proper (a) faultless (a) accurate (a) →	meticulous(a)
6. perfect →	correct (b^2) →	right (c) proper (a) faultless (a) accurate (a) →	meticulous (a)	
7. meticulous →	exact (a) →	correct (a) →	right (c) proper (a) faultless (a) accurate (a) perfect (a)	

FIGURE 7 (a)

CLASS SIX:INTELLECT

I.INTELLECTUAL FACULTIES AND P		
A.FACULTIES		
465	INTELLECT	41
466	INTELLIGENCE,WI	254
467	WISE MAN	48
468	UNINTELLIGENCE	34
469	FOOLISHNESS	22
470	FOOL	23
471	SANITY	205
472	INSANITY,MANIA	166
473	ECCENTRICITY	7
*****	SUM OF LEVEL 3	800
B.COMPREHENSION		
474	KNOWLEDGE	295
475	INTELLECTUAL	6
476	IGNORANCE	21
*****	SUM OF LEVEL 3	322
C.FUNCTIONS OF THE MIND		
477	THOUGHT	269
478	IDEA	117
479	ABSENCE OF THOU	1
480	INTUITION	42
*****	SUM OF LEVEL 3	429
D.REASONING PROCESSES		
481	REASONING	590
482	SOPHISTRY	48
*****	SUM OF LEVEL 3	638
E.CONSIDERATIONS		
483	TOPIC	234
484	INQUIRY	675
485	ANSWER	117
486	SOLUTION	194
487	DISCOVERY	139
*****	SUM OF LEVEL 3	1359
F.ASSESSMENT		
488	EXPERIMENT	979
489	MEASUREMENT	187
490	COMPARISON	59
491	DISCRIMINATION	113
492	INDISCRIMINATIO	1
*****	SUM OF LEVEL 3	1339
G.CONCLUSION		

	493	JUDGMENT	699	
	494	PREJUDGMENT	25	
	495	MISJUDGMENT	2	
	*****	SUM OF LEVEL 3	726	
H.THEORY				
	498	THEORY,SUPPOSIT	443	
	499	PHILOSOPHY	24	
	*****	SUM OF LEVEL 3	467	
I.BELIEF				
	500	BELIEF	710	
	501	CREDULITY	21	
	502	UNBELIEF	163	
	503	INCREDULITY	9	
	*****	SUM OF LEVEL 3	903	
J.GROUNDS FOR BELIEF				
	504	EVIDENCE,PROOF	752	
	505	DISPROOF	159	
	*****	SUM OF LEVEL 3	911	
K.QUALIFICATIONS				
	506	QUALIFICATION	516	
	507	NO QUALIFICATIO	91	
	508	POSSIBILITY	98	
	509	IMPOSSIBILITY	12	
	510	PROBABILITY	117	
	511	IMPROBABILITY	6	
	512	CERTAINTY	459	
	513	UNCERTAINTY	261	
	514	GAMBLE	142	
	*****	SUM OF LEVEL 3	1702	
L.CONFORMITY TO FACT				
	515	TRUTH	1058	
	516	MAXIM	193	
	517	ERROR	141	
	518	ILLUSION	70	
	519	DISILLUSIONMENT	6	
	*****	SUM OF LEVEL 3	1468	
M.ACCEPTANCE				
	520	ASSENT	668	
	521	AFFIRMATION	325	
	522	NEGATION,DENIAL	29	
	*****	SUM OF LEVEL 3	1022	
*****		SUM OF LEVEL 2	12086	10.38%

II.STATES OF MIND

A.MENTAL ATTITUDES				
	523	MENTAL ATTITUDE	378	
	524	BROAD-MINDEDNES	77	
	525	NARROWMINDEDNES	105	
	526	CURIOSITY	22	
	527	INCURIOSITY	5	
	528	ATTENTION	400	
	529	INATTENTION	25	
	530	DISTRACTION	24	
	531	CARE	162	
	532	NEGLECT	70	
	*****	SUM OF LEVEL 3	1268	
B.CREATIVE THOUGHT				
	533	IMAGINATION	87	
	534	UNIMAGINATIVENE	31	
	*****	SUM OF LEVEL 3	118	
C.RECOLLECTION				
	535	MEMORY	124	
	536	FORGETFULNESS	14	
	*****	SUM OF LEVEL 3	138	
D.ANTICIPATION				
	537	EXPECTATION	105	
	538	INEXPECTATION	26	
	539	DISAPPOINTMENT	33	
	540	FORESIGHT	9	
	541	PREDICTION	1	
	542	FOREBODING	74	
	*****	SUM OF LEVEL 3	248	
*****		SUM OF LEVEL 2	1772	1.52%
III.COMMUNICATION OF IDEAS				
A.NATURE OF IDEAS COMMUNICATED				
	543	MEANING	340	
	544	LATENCY	166	
	545	MEANINGLESSNESS	4	
	546	INTELLIGIBILITY	215	
	547	UNINTELLIGIBILI	103	
	548	AMBIGUITY	13	
	549	FIGURE OF SPEEC	2	
	550	INTERPRETATION	176	
	551	MISINTERPRETATI	3	
	*****	SUM OF LEVEL 3	1022	
B.MODES OF COMMUNICATION				
	552	COMMUNICATION	283	
	553	MANIFESTATION	386	
	554	DISCLOSURE	181	

555	INFORMATION	360
556	NEWS	116
557	PUBLICATION	656
558	COMMUNICATIONS	208
559	MESSENGER	12
*****	SUM OF LEVEL 3	2202

C.EDUCATION

560	TEACHING	579
561	MISTEACHING	3
562	LEARNING	114
563	TEACHER	18
564	STUDENT	63
565	SCHOOL	58
*****	SUM OF LEVEL 3	835

D.INDICATION

566	INDICATION	323
567	INSIGNIA	88
568	RECORD	372
569	RECORDER	59
*****	SUM OF LEVEL 3	842

E.REPRESENTATION

570	REPRESENTATION	182
571	MISREPRESENTATI	7
*****	SUM OF LEVEL 3	189

F.ARTS OF DESIGN

572	ART	142
573	SCULPTURE	30
574	CERAMICS	13
575	PHOTOGRAPHY	64
576	ENGRAVING	87
577	ARTIST	6
*****	SUM OF LEVEL 3	342

G.LANGUAGE

578	LANGUAGE	90
579	LETTER	54
580	WORD	69
581	NOMENCLATURE	116
582	ANONYMITY	4
583	PHRASE	129
*****	SUM OF LEVEL 3	462

H.GRAMMAR

584	GRAMMAR	313
585	UNGRAMMATICALNE	3
*****	SUM OF LEVEL 3	316

I.STYLE:MODE OF EXPRESSION			
586	DICTION	59	
587	ELEGANCE	75	
588	INELEGANCE	10	
589	PLAIN SPEECH	40	
590	CONCISENESS	22	
591	DIFFUSENESS	36	
*****	SUM OF LEVEL 3	242	
J.SPOKEN LANGUAGE			
592	SPEECH	451	
595	CONVERSATION	327	
597	ELOCUTION,PUBLI	57	
598	ELOQUENCE	129	
599	GRANDILOQUENCE	25	
*****	SUM OF LEVEL 3	989	
K.WRITTEN LANGUAGE			
600	WRITING	135	
601	PRINTING	177	
602	CORRESPONDENCE	147	
603	BOOK,PERIODICAL	363	
604	TREATISE	80	
605	COMPENDIUM	56	
*****	SUM OF LEVEL 3	958	
L.LINGUISTIC REPRESENTATION			
606	DESCRIPTION	260	
607	POETRY	39	
608	PROSE	33	
609	DRAMA	641	
610	ACTOR	31	
*****	SUM OF LEVEL 3	1004	
M.UNCOMMUNICATIVENESS:SECRECY			
611	UNCOMMUNICATIVE	25	
612	SECRECY	85	
613	CONCEALMENT	22	
*****	SUM OF LEVEL 3	132	
N.FALSEHOOD			
614	FALSENESS	256	
615	EXAGGERATION	28	
616	DECEPTION	84	
617	DECEIVER	32	
618	DUPE	14	
*****	SUM OF LEVEL 3	414	
*****	SUM OF LEVEL 2	9949	8.54%

TOTAL IN CLASS SIX:INTELLECT	23807	20.44%

CLASS SIX:INTELLECT Figure 7 (b)

I.	INTELLECTUAL FACULTIES AND P		
	A. FACULTIES	800	
	B. COMPREHENSION	322	
	C. FUNCTIONS OF THE MIND	429	
	D. REASONING PROCESSES	638	
	E. CONSIDERATIONS	1359	
	F. ASSESSMENT	1339	
	G. CONCLUSION	726	
	H. THEORY	467	
	I. BELIEF	903	
	J. GROUNDS FOR BELIEF	911	
	K. QUALIFICATIONS	1702	
	L. CONFORMITY TO FACT	1468	
	M. ACCEPTANCE	1022	
	***** SUM OF LEVEL 2	12086	10.38%
II.	STATES OF MIND		
	A. MENTAL ATTITUDES	1268	
	B. CREATIVE THOUGHT	118	
	C. RECOLLECTION	138	
	D. ANTICIPATION	248	
	***** SUM OF LEVEL 2	1772	1.52%
III.	COMMUNICATION OF IDEAS		
	A. NATURE OF IDEAS COMMUNICATED	1022	
	B. MODES OF COMMUNICATION	2202	
	C. EDUCATION	835	
	D. INDICATION	842	
	E. REPRESENTATION	189	

D. 2. TOWARD A STATISTICAL SUPPORT PACKAGE

by Frank Joyce

The long-term goal of the Automated Language Analysis project is to provide a system of programming packages which will allow a computerized analysis to characterize comprehensively the style of a non pre-edited natural language text. For this project, style is defined as "the patterns formed in the linguistic encoding of information." Such patterns include (but are not limited to) patterns of word choice, types of words, punctuation, and groupings of words and letters. Also to be taken into consideration are semantic patterns such as thematic development, physical location of semantically related words in the text, any contextual restraints which might be operating on the placement of related words, and the discovery of measurable correlates to stylistic concepts such as tone, vividness, and readability.

One of the guiding principles of this project has been the thesis that language not only functions to descrive some reality, but that it is, itself, an important reality meriting careful study. The importance of language as a behavior lies in the fact that "individuals and nations respond to the language of other individuals and nations with verbal and non-verbal actions which evoke other verbal and non-verbal actions" (Sedelow, 1969). Strong reinforcement for this position is provided by the noted psychologist, B. F. Skinner (Skinner, 1971). He argues that social scientists, including linguists, tend to become so engrossed in theories about the inner nature of the "autonomous individual" who is behaving that they neglect studying the behavior itself. It is an experimental question whether Skinner is correct in asserting that social scientists must abandon the "pre-scientific" concern with

mentalistic theories and concentrate instead on the actual behavior. In any case, the value of a comprehensive description of the style of language seems undeniable.

In view of the complexity of natural language, however, attaining such a comprehensive description is a massive undertaking. The first obstacle to be overcome is the sheer volume of data confronting the researcher. It is only in the last decade that the electronic computer has begun to provide the speed and information storage potential required to manipulate such large quantities of information. And even today, although a data base of computer accessible texts has begun to accumulate, the researcher often discovers that the most time-consuming and costly part of his task is getting even a non pre-edited text of any substantial size properly keypunched and proof-read. Once past that hurdle, he is still faced by the necessity of developing computational techniques for manipulating the data and extracting valuable information. This task is complicated by two factors. Firstly, the lack of a comprehensive package of computer analysis routines forces the researcher to retrace the costly steps of programming and debugging routines to accomplish tasks which other researchers have found valuable. Secondly, the researcher soon discovers that many of the more powerful analysis techniques of classical statistics are inapplicable in the face of data that refuse to conform to the assumptions of the normal probability distribution model, even after the employment of standard data transformation methods. Thus, stubborn data processing and mathematical problems must be solved before substantial progress can be made toward answering linguistic questions. The rest of this discussion will concentrate on progress made in developing a generalized statistical analysis package.

Last year's research report (Sedelow, 1970) included the results of theoretical investigations of the requirements for a general purpose statistical analyzer for natural language texts and the data base required by such an analyzer. It was decided that any such system should be subdivided into functional modules which would be under the control of general driver modules. Thus new functions could be easily added and new methods of interaction between modules could be more easily inplemented. This goal has been adopted in my investigation.

A review of the literature in the field of computational stylistics will establish that the use of frequency distributions is extremely common. Such distributions arise naturally when investigating characteristics such as letters per word, words per sentence, or frequency of word usage. These characteristics can be easily summarized by building a table or frequency distribution which shows how many words occurred only once in the text, how many words occurred just twice, and so on. The distribution can then be further summarized by taking measures of central tendency, such as the mean, or "average," number of letters per word, and measures of dispersion, such as the variance of standard deviation. The frequency distribution also provides a basis for more detailed analysis of variance. Therefore, the major effort to date has been directed toward investigating techniques for using frequency distributions.

The present procedure begins with a program which extracts frequency data from the SUFFIX output file which has been sorted back into text order. Information is extracted for variables such as letters per content word, letters per function word, letters per sentence, and

content words per sentence. In all, 34 such variables are measured (See Note 1, p. 61). Other programs extract data for variables such as frequency of word occurrence and position of function words and punctuation within sentences. The raw variates are then processed by a program which groups them into frequency distributions, calculates basic statistics, tests for normality on the basis of the coefficients of skewness and kurtosis as explained below, performs a variety of transformations of the data--again testing for normality--and produces a histogram of the distribution. When this program has been run on a number of different samples, the frequency distributions can be run through another program which lists the basic statistics in tabular form for easy comparison.

The text samples used were purposely diverse, both in terms of size and of subject matter. The project has available two different translations from the Russian of a book entitled Soviet Military Strategy. Since the subject matter is precisely identical for both versions, it is reasonable to assume that the only uncontrolled variable is the style of the translators. Hence, the first two text samples (hereafter referred to as SMS 1 and SMS 2) consist of the first chapter of each translation. The sample sizes are 17,979 and 17,916 words respectively. Next, two samples were taken from James Joyce's The Dubliners. The samples are the entire story, "The Sisters" (referred to as JOYCE TS; N = 3043), and a long passage taken from the middle of "The Dead" (JOYCE TD; N = 5242). Finally, five samples were taken from the works of Edgar Allen Poe. They are the short stories "The Tell Tale Heart" (POE 1; N = 2143) and "The Cask of Amontillado"

(POE 2; N = 2329) and the three poems "Lenore" (POE 3; N = 111), "To Science" (POE 4; N = 86), and "The Raven" (POE 5; N = 1065).

The first question to be answered deals with how closely the data fits the assumptions underlying the normal probability distribution model. The normal probability distribution is a special type of symmetrical distribution. The question of which measures of central tendency and dispersion best summarize the distribution is easily answered since the shape of the distribution is completely determined by two parameters, the mean and the variance. The distribution is particularly easy to analyze mathematically and a variety of elegant and powerful statistical techniques have been developed for analyzing data which can meet the criteria of normality. The normal distribution is based on the assumption of an indefinitely large population which can take on values indefinitely far away from the mean in either direction with indefinitely small gradations of value. In contrast, consider the distribution of a characteristic such as word length. It is logically impossible for a word to have less than one letter and in every language there is, in practice, a finite upper limit to the number of letters per word. Also, gradations in value take place in discrete jumps; for example no actual word consists of 2.5 letters. At best then, such a characteristic can only approximate normality. Even this approximate normality is unlikely, however, since the normal distribution is derived mathematically by assuming that the values of the variable are the result of a great number of elementary causes which independently combine in an additive fashion. Thus, individual factors tend to cancel each other out resulting in relatively few extreme values and a symmetric distribution about the mean. On the other hand, language characteristics

tend to have skewed distributions, that is, there are considerably more extreme values in one direction than the other. For example, in looking at letters per paragraph in JOYCE TD we find that the shortest paragraph contained 10 letters and the longest paragraph contained 1284 letters. The distribution reveals that fully half of the paragraphs are contained in the interval from 10 to 26 letters. Ninety-five percent of the paragraphs contain 141 or fewer letters, leaving five percent of the variates spread in the interval from 141 to 1284. Thus, the distribution is highly positively skewed.

The tests for normality of the obtained distributions were based on measures of skewness and kurtosis of the distribution. As was mentioned above, skewness is the degree of departure from symmetry. Kurtosis is the degree of peakedness or pointedness of the distribution. The particular measurements used were the moment coefficients[1] of skewness and kurtosis, both of which would equal zero if the distribution were normal. In the example from JOYCE TD mentioned above, the coefficient of skewness was 8.24, indicating large positive skewness as expected. The coefficient of kurtosis was 87.01, indicating a very peaked distribution. Table 1[2] shows these two coefficients for several characteristics for JOYCE TD and for SMS 1. All coefficients depart from normality significantly at the .005 level, based on the t-test.

---

1. Let $X_1, X_2, \ldots, X_k$ occur with frequencies $f_1, f_2, \ldots, f_k$, respectively. Then the mean, $\overline{X}$, is given by $\overline{X} = \frac{\Sigma fX}{N}$. The rth moment about the mean X is defined as $m_r = \frac{\Sigma f(X-\overline{X})}{N}$. The moment coefficient of skewness is $a_3 = \frac{m_3}{s^3}$, where $s = \sqrt{m_2}$ is the standard deviation. The moment coefficient of kurtosis is $a_4 = \frac{m_4}{s^4} - 3$.

2. Tables appear on pp. 63-65.

Since the distributions do not approximate normality, the next step is to transform the data in such a way that the transformed distribution is normal. Recall that the normal distribution results from the additive effect of a number of small causes. Now, if a = b x c x d, recall that log a = log b + log c + log d. Thus, while a is the product of three other values, log a is the sum of three other values. This suggests that the logarithms of the variates from a distribution resulting from multiplicative interactions would be the result of additive interactions; and, therefore, the logarithms might be normally distributed. Such a distribution is known as a lognormal distribution. Lognormality has been hypothesized for word length (Herdan, 1966). This lognormality is suggested as the optimal distribution which natural languages approximate to various degrees of closeness. Herdan plotted the variates on log probability graph paper and the plots were sensibly straight lines, thereby admitting the hypothesis of lognormality. This study tested the lognormality assumption by transforming the variates and measuring skewness and kurtosis. The results were negative in all cases. In some cases either skewness or kurtosis was reduced sufficiently to admit the normality hypothesis, but in no case were both satisfactorily altered. Thus, we must conclude that the hypothesized optimalization process, if valid, is not sufficiently reflected in the samples tested to allow statistical procedures to be based on the lognormal assumption. Table 2 shows the results of the logarithmic transformation on the characteristics listed in Table 1.

In working with word frequencies, Carroll (Carroll, 1967, 1968) transformed the word probabilities, that is, the frequencies of occurrence

divided by the sample size. Carroll reported positive results, again based on the graphic method. Carroll also suggested that allowances be made for the effect of sample bias. This arises due to the fact that in sampling from a large population, we are more likely to pick words which occurr frequently in the population. On the other hand, words which occur only rarely in the population will be missed by the sampling procedure. The implications of these findings will be examined during the coming year. However, Carroll's best results were for samples which were substantially larger than those of this study; therefore, the practical validity of the lognormal assumption must still be considered highly problematical, with the greatest hope for success lying with word frequency applications.

In addition to the logarithmic transformation, a number of other transformations were applied. The square roots of the variates (e.g., number of letters per word) were taken, the logarithms of both the variates and the frequency with which the variates occurred in the samples were also taken, and the logarithm of the frequencies were taken leaving the variates untransformed. The results in all cases were completely negative at the .005 probability level.

Apparently, then, statistical analyses of style are going to require techniques that do not rely on the assumption of normality. The implications of this fact on making valid inferences from a sample to a larger population will be discussed below. The process of calculating statistics which describe a sample is not so drastically altered, although one immediate result is that the mean and variance are no longer automatically the preferred measures of central tendency

and dispersion respectively. Firstly, they no longer completely determine the shape of the distribution. Secondly, it has been shown that the distributions tend to be highly skewed, with the bulk of the variates forming a fairly compact, sharp peak accompanied by a relatively small number of extreme values spread out to the positive side of the peak. It is precisely this kind of distribution that the mean and variance most poorly represent, since they give relatively great weight to extreme cases. A better way to summarize these distributions is in terms of percentiles, those values dividing the data into one hundred equal parts. Thus, five percent of the data lie below the 5th percentile. The 50th percentile is the middle value (or the mean of the two middle values) and is called the median. As will be seen, the median is a more representative measure of central tendency for skewed distributions than is the mean.

In this study, the definition of the percentile has undergone a minor revision. For example, if we were given the lengths of twenty-one words distributed as in Table 3.1, then since the median is the middle value (in this case the 11th value) the median would be 4 since the 11th value is 4. If we were to add 10 words of length six in Table 3.2, the median would then be the 16th value which is still 4. Thus, the median would fail to reflect the difference between the two distributions. In this study, percentiles are calculated by assuming that the variates are evenly spread along the interval for one-half unit on either side of the actual value. Therefore, the ten variates with value 4 would be considered to be evenly spread out along the interval from 3.5 to 4.5. Then since the 11th value is the fourth

value in this interval, the median for distribution 3.1 would be 3.9. The 16th value is the ninth value in the interval and so the median for distribution 3.2 would be 4.4. While it is clear that no word can have 4.4 letters, it is also clear that the central tendency is more accurately represented by allowing this modification.

In order to demonstrate the superiority of the median over the mean when the distribution is skewed, we again refer to the example of letters per paragraph in JOYCE TD. The mean number of letters per paragraph was 49.91, or about 50, while the median number of letters was only 26.09, or about 26. Indeed, over three-fourths of the paragraphs had 56 letters or fewer. A similar situation exists for SMS 1 where the mean is 114.36 letters, the median is 77.88, and the 75th percentile is 138.33. It can be seen that the mean is closer to the 75th percentile than the center of the distribution. Of course, great fluctuation is to be expected in a characteristic like letters per paragraph; but even when the median and mean do not dramatically differ, the loss of the normality assumption deprives the mean of its automatic claim to superiority, thus the median is representative in general whereas the mean is only sometimes so.

As an example of how percentiles may be used to describe distributions, we now consider Tables 4 and 5 summarizing the distributions for length of content words and length of function words for the samples. It can be seen from Table 5 that the function word distributions for all samples are extremely similar. This is completely consistent with previous studies which have also indicated that function word length is more controlled by the rules of language than by style considerations.

On the other hand, in Table 4 the two SMS samples show a tendency toward longer content words than the fiction and poetry samples. This is consistent with other studies which have shown a similar difference between scientific or technical works and works of fiction.

The above results are in agreement with previous efforts. However, at this point any inferences about a larger population of language based on results in this study would be premature. The inability to assume normality greatly complicates the process of making statistical inferences. For example, it is not clear what constitutes a valid sampling technique. Various researchers have advocated an assortment of techniques. According to one review paper (Posner, 1963) Yule advocated "spread sampling": that is, spreading the sample as uniformly as possible over the entire text. This might be done by selecting a number of words, sentences, lines or paragraphs from each page. A discussion of word categories which should be included is found in Herdan (1966). He points out that a number of investigators have excluded verbs and function words on the grounds that stylistic differences are best brought out by nouns and, to a lesser degree, by adjectives. However, this is highly dependent on the definition of style used, since nouns are also highly sensitive to subject matter. Thus, in distinguishing between the styles of various genres of writing one might use different criteria than would be used to distinguish the individual style of a particular writer over a broad spectrum of genres. The implications of each of these approaches with respect to problems of sampling error are not clear.

A further problem arises in determining sample size. Various researchers have advocated sizes ranging from 4,000 to 50,000 words

(Posner, 1963; Milic, 1967). In most applications of statistics, the larger the sample, the less chance for error. However, in sampling for vocabulary diversity, large samples can lead to distortion by increasing the probability of selecting more hapax legomena (words occurring just once) which will distort the proportions (Posner, 1963). Thus, for this application medium sized samples might be preferable (but what is "medium sized"?). When the population to be sampled fits the normality assumptions, one can calculate how large a sample is required to hold expected sampling errors within a pre-specified range. However, as has been shown, this expedient has been denied us. Now that we have developed the beginnings of a statistical package, we can begin to design experiments which will make clear what effects result from altering sample size and selection techniques. As an example, we will soon be able to partition large texts into segments and obtain the distributions for characteristics that have already been measured for the whole text. We can then examine fluctuations throughout the text. Those characteristics which remain stable throughout the body of a text can be discovered. Also, the effect of segment size on stability of the distribution will be studied. Hopefully we will be able to provide some solid information on which to base sampling procedures.

In addition, we will explore statistical techniques which are not based on the normality assumption. A number of non-parametric methods commonly used in areas such as biology and psychology have already been investigated. They include runs tests such as the Wald-Wolfowitz test, the Mann-Whitney or Wilcoxen test which may be used in situations where a runs test is appropriate, and the Kolmogaorov-Smirnov test. All of these tests may be used to determine whether different samples come from

the same population. However, as is common with most prominent nonparametric tests, it is assumed that the individual variates are independent, that is the value of one variate does not influence the value of another variate. It seems highly unlikely that word lengths, for example, do not actually exert influence upon each other. Another common requirement in nonparametric tests is that the data be free from ties, that is, the variates should not have equal values. A glance at Table 5 will demonstrate how badly this requirement is generally violated. Whether efficient tests may be developed which do not require statistical independence and permit ties remains to be seen.

Another area to be explored is the representation of characteristics which alter from one part of the text to another. By summarizing variates into frequency distributions, the temporal nature of language is being neglected. We will attempt to develop techniques for uncovering sequential style traits. For example, two authors might be very similar with respect to number of words and sentences per paragraph. However, one author may prefer to open and close paragraphs with short sentences and use long sentences in the body of the paragraph. The other author might use sentences of uniform length throughout the paragraph. Finally, we will investigate the utility of statistics in the area of content analysis. As an example, the program which derives frequency distributions for segments of the same text may also be used to trace thematic shifts by using information from program THESAUR. Thus, it might be possible to characterize segments of text on the basis of thematic content.

In her review of work in stylistic statistics, Rebecca Posner (Posner, 1963) is struck by the fact that she has shown that very little has been done in the field. After spending several months encountering

one blind alley after another, it has become clear to us why this may be true. Research in this area requires large expenditures of time and computer resources and is characterized by high risk due to a lack of techniques which are adequate for the task. Hopefully by mounting a sustained effort using the programs already developed by this project as a starting point we will be able to make substantial progress in this area.

Note 1: VARIABLES TESTED

Frequency data were collected for the following variables (Note: "Clauses" were operationally defined as those words preceding a punctuation mark. It is recognized that this does not coincide very closely with the linguistic entity known as a clause. However, by taking into consideration conjunctions, this definition may be easily expanded to approximate the linguistic entity more closely):

Letters per content word; Letters per function word; Letters per word (the sum of the previous two mutually exclusive categories); Letters per clause; Letters per sentence; Letters per paragraph; Content words per clause; Content words per sentence; Content words per paragraph; Function words per clause; Function words per sentence; Function words per paragraph; Clauses per sentence; Clauses per paragraph; Sentences per paragraph; Frequency of occurrence of function words; Frequency of occurrence of content words (not grouped together by SUFFIX); and, Frequency of occurrence of content word root groups (grouped together by SUFFIX).

In addition, separate counts were kept for the number of function words and content words occurring in clauses ended by each of the

following punctuation marks: comma, semicolon, colon, ellipse, dash, period, question mark, and exclamation point.

REFERENCES

Carroll, John B. "On Sampling from a Lognormal Model of Word-frequency Distribution" in Computational Analysis of Present-day American English by Henry Kucera and W. Nelson Francis. Brown University Press, Providence, Rhode Island, 1967.

_____. "Word-frequency Studies and the Lognormal Distribution" in Proceedings of the Conference on Language and Language Behavior. Eric M. Zale, ed. Appleton-Century-Crofts, New York, 1968.

Herdan, Gustav. The Advanced Theory of Language as Choice and Chance. Springer-Verlag, New York, 1966.

Milic, Louis Tonko. A Quantitative Approach to the Style of Jonathan Swift. Mouton, The Hague, 1967.

Posner, Rebecca. "The Use and Abuse of Stylistic Statistics" in Archivum Linguisticum, Volume 15, Fascicule 2 (1963), pp. 110-139.

Sedelow, Sally Yeates. Automated Language Analysis, Report on research for the period March 1, 1968 to February 28, 1969, Contract N000 14-67-A-0321, Office of Naval Research, University of North Carolina. DDC # AD 691-451.

Skinner, B. F. Beyond Freedom and Dignity. Alfred A. Knopf, New York, 1971.

TABLE 1

Coefficients of Skewness and Kurtosis

	JOYCE TD			SMS 1		
	Skewness	Kurtosis	N	Skewness	Kurtosis	N
Letters per Content Word	1.15	2.36	2493	0.69	1.17	9660
Letters per Function Word	1.31	3.36	2749	2.08	8.52	8319
Letters/Word	1.14	1.94	5242	0.88	0.46	17979
Frequency of Word Usage (Grouped by matchcount)	8.04	91.30	1028	13.63	250.49	2059

All values are significant at the .005 level.

TABLE 2

Coefficients of Skewness and Kurtosis
for Logrithmically Transformed Data

	JOYCE TD			SMS 1		
	Skewness	Kurtosis	N	Skewness	Kurtosis	N
Letters per Content Word	0.22*	2.36*	2493	-0.33*	0.07	9660
Letters per Function Wordl	0.05	3.04*	2749	0.53*	7.92*	8319
Letters/Word	0.04	0.85*	5242	0.06*	-0.44*	17979
Frequence of Word Usage (by matchcount)	2.12*	5.61*	1028	1.26*	2.01*	2059

*Value is significant at the .005 level.

TABLE 3.1

Hypothetical Distribution

Word Length	Number of Words
1	0
2	2
3	5
4	10
5	4
	N = 21

TABLE 3.2

Hypothetical Distribution

Word Length	Number of Words
1	0
2	2
3	5
4	10
5	4
6	10
	N = 31

TABLE 4

Letters Per Content Word

Samples	V	LOW	HIGH	MEAN	Percentiles 0.05	0.25	0.50	0.75	0.95	Coefficient of Variance
JOYCE TD	2493	1	17	5.708	3.112	4.192	5.382	6.813	9.451	35.115
JOYCE TS	1292	2	15	5.742	3.046	4.108	4.322	6.961	9.659	36.730
SMS 1	9660	2	18	7.445	3.448	5.547	7.320	9.054	12.044	35.873
SMS 2	9427	2	18	7.445	3.538	5.575	7.341	9.025	11.964	35.042
POE 1	892	2	14	5.661	2.898	4.091	5.366	6.893	9.376	35.710
POE 2	1003	1	15	6.127	2.996	4.277	5.747	7.562	10.527	37.846
POE 3	55	3	9	5.564	3.417	4.132	5.182	6.925	8.812	30.699
POE 4	47	2	12	5.489	2.950	3.984	4.937	6.562	9.325	38.150
POE 5	548	3	11	5.880	3.551	4.307	5.727	7.126	9.200	30.317

TABLE 5

Letters Per Function Word

Samples	V	LOW	HIGH	MEAN	Percentiles 0.05	0.25	0.50	0.75	0.95	Coefficient of Variance
JOYCE TD	2749	1	10	3.006	1.277	2.176	2.880	3.471	5.306	41.119
JOYCE TS	1751	1	10	2.920	1.139	2.044	2.780	3.440	5.235	43.494
SMS 1	8319	1	12	2.923	1.534	2.077	2.741	3.383	5.219	42.037
SMS 2	8489	1	12	2.880	1.536	2.045	2.701	3.339	5.139	41.755
POE 1	1251	1	10	2.844	0.877	1.942	2.787	3.492	5.065	43.855
POE 2	1326	1	12	2.762	0.879	1.874	2.654	3.380	5.112	45.467
POE 3	56	1	6	3.125	1.664	2.574	3.093	3.700	4.447	28.005
POE 4	39	1	5	2.897	1.475	2.205	2.882	3.456	4.525	32.448
POE 5	517	1	9	3.029	1.062	2.037	2.883	3.843	5.372	43.808

D. 3. STATISTICAL ANALYSIS OF LINGUISTIC FREQUENCY DATA[1]

by Harrel Wright

INTRODUCTION

The primary objective of this paper is to present a statistical bibliography which may be of use to computational linguists. The secondary objective is to present a brief discussion of some of these techniques and their relative merits.

The techniques presented here were selected primarily because of their possible use in the analysis of frequency data--data about the frequency of words in a text or the frequency of equivalence classes established over the words in a text. Some of the techniques may prove useful in the analysis of positional data--data about the position of words or equivalence classes within a unit of length or structure.

The complexities of analyzing positional data will undoubtedly be reflected in the difficulty of the statistical techniques required. Some sections of this paper and the bibliography are provided with the analysis of positional data in mind.

BIBLIOGRAPHY

The items of the bibliography have been selected with regard to the general scope of the item and the number of practical, detailed examples; Kendall and Stuart (1967, 1968, and 1969) being the work which best fulfills the first criterion, and Sokal and Rohlf (1969) being the best example of the second.

I have not selected items with their specific value to computational linguistics in mind because I am not a linguist, computational

or otherwise. I have tried to avoid any assumptions about the nature of the distribution which might produce observed linguistic data. Such assumptions seem unwarranted, even in the asymptotic case.

GENERAL CONSIDERATIONS

When dealing with experimental data, the statistician first tries to determine the nature of the distribution which underlies that data. Second, once the nature of the distribution is known, he tries to normalize the data and employ well-known, more-or-less traditional techniques, or to use techniques based upon the actual underlying distribution. If the nature of the underlying distribution cannot be determined, then distribution-free techniques may prove useful.

During the summer[2], attempts to normalize the frequency data from a text yielded doubtful results. Determination of the underlying distribution is difficult and may not be worth the effort.

The use of contingency table analysis seems to be the most useful; that is, it is easy to use, requires few assumptions, and assigns a number to the results. Contingency table analysis holds the most promise for further analysis of frequency data. This technique will contribute little to the analysis of positional data, however.

In addition, the underlying distribution seems to have an interesting asymptotic behavior. Linguistic analysis involves relatively large sample sizes and this should have proved a mixed blessing. Large N restricts, makes inconvenient, the use of distribution-free techniques. On the other hand, large N <u>should</u> facilitate the asymptotic approach of the underlying distribution toward the normal distribution and the use of traditional techniques. This does not appear to be the case.

With relatively large N, the distribution which underlies the frequency data of a text does not appear to be normal, nor does it approach normal in any useful way, nor is it easily tranformed to the normal. There seems to be a constraint in the asymptotic behavior, in that words with few letters have a higher frequency of occurrence than words with more letters.

Large sample size, however, is beneficial to the use of contingency tables. It may also prove helpful in the use of binomial "success-failure" ("hit-or-miss") comparison of populations studies.

Large sample size also hinders the analysis of positional data. The difficulties in the analysis of a transition matrix of a Markov process predicting word transitions should prove considerable with the great number of words or classes of words, and the position information necessary in linguistic analysis.

PARAMETRIC APPROACHES

This section is concerned with statistical approaches which make use of the normal distribution in some way. As such standard techniques as analysis of variance, t-test, and chi-squared goodness-of-fit tests are well documented, perhaps only a brief discussion of standard references and normalizing transformations is required.

Easy-to-intermediate sources of information about these techniques are Sokal and Rohlf (1969), Meyer, and Hays. (Meyer, and Hays refers to two textbooks predominantly concerned with psychology. Biometry is a beautiful book with many examples, worked out in full detail; the biometrical nature of these examples may be ignored or savored, according to the reader.) Advanced practical and theoretical

information may be obtained from Winer (1962), Scheffe (1959), and Kendall and Stuart (1967, 1968, 1969).

Transformations. Primarily, the following transformations are used in the problem of heterogenity of variance in analysis of variance. However, they are also known as normalizing transformations.

The SQUARE-ROOT transformation is applied when the cell distributions are Poisson; that is, when treatment means and variances are proportional. There is a correction factor for small scores but this does not appear to be an important consideration in the linguistic analysis of a text.

The ARC SINE transformation is applied when the cell distributions are binomial in form; that is, generally, when the scores are proportions.

The LOGARITHMIC transformations are used when the cell standard deviations are proportional to the cell means; generally, in this case, the distributions are positively skewed.

These transformations are discussed in the easy-to-intermediate sources mentioned above. Bartlett (1942) is the source article for these works as well as an excellent discussion of the use of the transformation. Odeh and Olds (1959) as well as Olds, et. al. (1956) provide information about the standard use of transformations.

If the distribution is known and non-normal, the problem of hypothesis testing becomes more involved. In addition to comparing parameters or populations, the related probabilities must be calculated. Statistical techniques based upon these non-normal distributions may be found in the general reference works mentioned above and in the bibliography.

Special note should be made of Haight (1967) and Lancaster (1969). Haight, a traffic analyst, has compiled a book about the Poisson distribution. Lancaster's book is, in many--not all--ways, a fundamental work about the chi-squared distribution and its applications. Lancaster's work has many good examples and references.

NONPARAMETRIC APPROACHES

The two most promising distribution-free techniques are contingency table analysis and the analysis of ranks. Contingency tables hold great promise for the comparison of populations via frequency data. Rank techniques may not be as useful but they are too powerful a tool to be ignored at this initial stage.

Contingency tables. Contingency tables will certainly be of use in the analysis of linguistic frequency data. The technique employs no assumptions about the underlying distribution--that is, none that are significant at any practical level. Large sample sizes are beneficial to their use, theoretically, although not necessarily to the calculations involved. The underlying distributions need not be symmetric or continuous.

The test statistic is well approximated by the chi-squared distribution for which many good tables exist. Yate has developed a correction which compensates (?) for the continuous approximation of the discrete test statistic by the chi-squared distribution. However, Yate's correction has been found overly conservative and its use is not recommended.

Sokal and Rohlf (1969) treat the technique well but I prefer the treatment of Lancaster (1969). Lancaster's examples seem better

presented although he presents the theory at a higher level (easily ignored). I did not notice the presence of Lancaster's earlier--and much disputed--interpretation of an interaction statistic.

Goodman and Kruskal (1954, 1959, 1963) offer a comprehensive treatment of the various test statistics. Bartlett (1935) offers another source article. Kullback's (1959) book on information theory has a chapter on contingency tables and the theory of the G-statistic.

In order to use the technique it need only be assumed that 1) samples are independent, 2) samples are random, and 3) the classes into which the data is divided are all-inclusive. For the sake of approximation, restrictions about the cell sample size proportions are often made. In general, such restrictions apply when cell sizes are less than, say, ten and therefore do not apply to the analysis of most texts.

RANK TECHNIQUES

I mention rank techniques because I think they may be useful in the analysis of positional data. Their usefulness in the analysis of either frequency or positional data is limited by the large sample size, relatively large numbers to which ranks will be assigned, and the certainty of many ties.

Rank techniques came about when someone prepared tables for Fisher's method of randomization applied to ranks rather than numbers for a particular problem. After assigning ranks to the numbers in a problem, one had access to universally general tables.

Conover (1971) has listed many nonparametric techniques in a useful, cookbook-like book. Bradley's (1968) work is equally good

at the level of intermediate theory. For advanced theory, Hájek and Sidák (1967) should be consulted.

For rank correlation techniques Kendall (1970) is probably best. In any event, when in doubt, use Kendall's tau over Spearman's rho.

GENERAL REFERENCE

Aitchison, J. and J. A. C. Brown. 1957. The Log-Normal Distribution. Cambridge: Cambridge University Press.

Bradley, J. V. 1968. Distribution-free Statistical Tests. Englewood Cliffs, New Jersey: Prentice-Hall.

Conover, W. J. 1971. Practical Nonparametric Statistics. New York: John Wiley.

Haight, F. A. 1967. Handbook of the Poisson Distribution. New York: John Wiley. (Publications in Operations Research, No. 11)

Kendall, M. G. and A. Stuart. 1969. The Advanced Theory of Statistics, Vol. I: Distribution Theory. New York: Hafner.

_____. 1967. The Advanced Theory of Statistics, Vol. II: Inference and Relationship. New York: Hafner.

_____. 1968. The Advanced Theory of Statistics, Vol. III: Design and Analysis; and Time Series. New York: Hafner.

Lancaster, H. O. 1969. The Chi-squared Distribution. New York: John Wiley.

Scheffé, H. 1959. The Analysis of Variance. New York: John Wiley.

Sokál, R. R. and F. J. Rohlf. 1969. Biometry, The Principles and Practice of Statistics in Biological Research. (Books in Biology Series) San Francisco: W. H. Freeman.

Winer, B. J. 1962. Statistical Principles in Experimental Design. New York: McGraw-Hill.

BAYESIAN STATISTICS

Edwards, W., H. Lindman, and L. J. Savage. 1963. Bayesian Statistical Inference for Psychological Research. Psych. Review., 70, 193-242.

Lindley, D. V. 1965. Introduction to Probability and Statistics From a Bayesian Viewpoint. 2 volumes. New York: Cambridge University Press.

Raiffa, H. 1968. Decision Analysis. Reading, Massachusetts: Addison-Wesley.

Zellner, A. 1971. Bayesian and Non-Bayesian Analysis of the Log-Normal Distribution and Log-Normal Regression. Journal American Statistical Association, 66, 327-330.

CONTINGENCY TABLES

Bartlett, M. S. 1935. Contingency table interactions. J. R. Statist. Soc., supp. 2, 248-252.

Goodman, L. A. 1963. On methods for comparing contingency tables. J. R. Statist. Soc. A, 126, 94-108.

_____. 1971. The Analysis of Multidimensional Contingency Tables: Stepwise Procedures and Direct Estimation Methods for Building Models for Multiple Classifications. Technometrics, 13, 33-61.

_____ and W. H. Kruskal. 1954. Measures of association for cross classifications. Journal American Statistical Association, 49, 723-764.

_____. 1959. Measures of association for cross classification. II: Further discussion and references. Journal American Statistical Association, 54, 132-163.

_____. 1963. Measures of association for cross classification. III: Approximate sampling theory. Journal American Statistical Association, 58, 310-364.

Ku, H. H., R. N. Varner, and S. Kullback. 1971. On the analysis of multidimensional contingency tables. Journal American Statistical Association, 66, 55-64.

Mosteller, F. 1968. Association and estimation in contingency tables. Journal American Statistical Association, 63, 1-28.

Plackett, R. L. 1969. Multidimensional contingency tables: A survey of models and methods. Bull. Intern. Stat. Inst., Book 1, 43, 133-142.

TRANSFORMATIONS

Bartlett, M. S. 1942. The use of transformations. Biometrics, 3, 39-52.

Odeh, R. E. and E. G. Olds. 1959. Notes on the analysis of variance of logarithms of variances. WADC tech. Note 59-82, Wright-Patterson AFB, Ohio.

Olds, E. G., T. B. Mattson, and R. E. Odeh. 1956. Notes on the use of transformations in the analysis of variance. WADC tech. Rep. 56-308, Wright Air Development Center.

INFORMATION THEORY

Attneave, F. 1959. Applications of Information Theory to Psychology. New York: Holt, Rinehart and Winston.

Goldman, S. 1953. Information Theory. (New York: Dover, 1968)

Kullback, S. 1959. Information Theory and Statistics. New York: John Wiley. (New York: Dover, 1968)

MARKOV PROCESSES

Anderson, T. W. and L. A. Goodman. 1957. Statistical inference about Markov chains. Ann. Math. Statist., 28, 89-110.

Billingsley, P. 1961a. Statistical Inference for Markov Processes. Chicago: University of Chicago Press.

_____. 1961b. Statistical methods in Markov chains. Ann. Math. Statist., 32, 12-40.

Gold, R. Z. 1963. Tests auxiliary to Chi-squared tests in Markov chains. Ann. Math. Statist., 34, 56-74.

Goodman, L. A. 1959. On some statistical tests for m-th order Markov chains. Ann. Math. Statist., 30, 154-164.

Kullback, S., M. Kupperman, and H. H. Ku. 1962. Tests for contingency tables and Markov chains. Technometrics, 4, 573-608.

RANK TECHNIQUES

Hájek, J. and Z. Sidák. 1967. Theory of Rank Tests. New York: Academic Press.

Hollander, M. 1963. A nonparametric test for the two-sample problem. Psychometric, 28, 395-403.

Jacobson, J. E. 1963. The Wilcoxon two-sample statistic: Tables and bibliography. J. Amer. Statist. Ass., 58, 1086-1103.

Kendall, M. G. 1970. Rank Correlation Methods. London: Griffin.

Laubschen, M. F., F. E. Steffens, and E. M. DeLange. 1968. Exact critical values for Mood's distribution-free test statistic for dispersion and its normal approximation. Technometrics, 10, 497-508.

Mood, A. M. 1954. On the asymptotic efficiency of certain non-parametric two-sample tests. Ann. Math. Statist., 25, 514-522.

NOTES

1. This paper as well as the rest of my part of the project was supported by the Office of Naval Research project for which S. Y. Sedelow is principal investigator.

The work on the contingency table overlaps work was supported by a Public Health Traineeship in Experimental Psychology, fund number 1811-9706-6 from the Psychology Department of the University of Kansas.

2. This is in reference to my brief exposure to practical computational linguistics; that is, numbers. Frank Joyce has actually employed statistical techniques in his analysis and comparison of texts.

D. 4. TOWARD A THEORY OF PREFIXING

by Sam Warfel

PREFIX is a part of the root-grouping portion of VIA which recognizes prefixes on the basis of comparison with words on an inclusion-exclusion (CLUD) list. Although the program efficiently handles large bodies of data within a reasonable amount of time, there are a number of problems associated with the nature of prefixes, themselves, which have not been dealt with theoretically, much less operationally. Thus, any current computer-based program dealing with prefixes poses dilemmas for which any given resolution markedly and perhaps incorrectly affects the semantic import of the words in question. To begin with, there is no model to serve as a guide for the selection of prefixed and unprefixed words for the inclusion-exclusion (CLUD) lists upon which PREFIX is based. The Random House Dictionary and experience were the informal guides for the selection process; no formal guides have been established. What is the basis for saying that pre- is not a prefix in preach, but is a prefix in prejudge? How should forms such as tele- and bio- be dealt with? Another type of problem, which affects the recognition of prefixes as well as that of the denotation of a given word's use, is that of the homograph. When is present a gift and when is it something which was pre-sent [sent before]? In order to construct a model or models which will help us deal with the first problem, a more rigorous investigation must be made of the prefixing process in English than has been undertaken heretofore. In order to deal with the second problem, the context in which the word occurs must be taken into account. It may be that some form of THESAUR, operating with a thesaurus,

will help define the context in which a given meaning of a word is most likely to be used.

This paper will primarily concentrate upon research investigations which might lead to a model of prefixing in English as well as perhaps, ultimately, in some other languages. All suggestions should be considered tentative because my research has been underway for just a few months.

The ideal for an operating PREFIX will thus be an algorithm based upon rules which are the substance of a prefix model. The three sources of information from which such a model could be constructed appear to be: 1) spelling, 2) phonetics, and 3) semantics. It is immediately obvious that, aside from conventions like the hyphen and apostrophe, the spelling system of English alone does not provide enough information for the proper assignment of any morpheme boundaries including prefixes. The problem of distinguishing proper prefixes in pairs like readjust-reading led to the present formulation of PREFIX. If phonetic information is available through a dictionary look-up some accuracy is gained. For example, rules which take account of 1) voicing, 2) syllable division, and 3) stress placement could recognize prefixes in cases not possible with spelling alone. The undesirability of considering pre in present as a prefix can be eliminated by indicating in the program that the voicing of the /s/ intervocalically precludes the existence of a morpheme boundary in the word.*

---

*The decision must be text-specific since present can have the meaning [sent before], in which case the /s/ is not voiced.

Syllable division would make possible the differentiation of postexilian (post-exilian) with a prefix, from posterior (pos-terior) without a prefix, and stress placement would indicate that realtor (réaltor) is not prefixed while realter (reálter) is. However, the usefulness of these phonetic indicators is extremely limited, so much so, that the examples given are practically the only ones found during a rather extensive dictionary search. In addition the rules needed to utilize the information given above would appear to be so complex as to vitiate any usefulness phonetics might have. It appears after considerable work that there is no system which will deal with words like the following using only spelling and phonetic information: predispose, prefix, preoccupied, presuppose, debark, deplane, deactivate, define, defile, to list only a few. For example, in the case of preoccupied there is no phonetic or spelling information which indicates whether the word is to be considered prefixed or not, since in this case it may be either, as I will discuss in a moment.

Since prefixes carry a greater semantic load than functional load it is not too surprising that semantics offers the greatest promise for prefix recognition. It is because of this semantic load that text-specific prefix analysis requires some contextual semantic information. Notice for example that it is necessary to know which sense of the word is meant in determining whether the word should be considered to have a prefix in the following cases: prescience--[before science] or [foreknowledge], preoccupied--[lost in thought] or [already occupied], presuppose--[imply] or [suppose beforehand].

Semantic information is likewise necessary for the proper analysis of non-text-specific prefixed words. It is not enough to match letters

from the beginning of the word with that of a prefix and then check to see if the remaining letters match that of a word in the language. This type of analysis would consider pre in prevent as a prefix since there is a word vent in English. However, the meaning of vent in prevent has no obvious semantic relationship with the meaning of prevent (unless of course the context indicates a reading of [vent beforehand]).

The problem can best be understood in terms of form and content. Presuming that there are a limited number of semantic primes which function in some cognitive grammar, the rules of the language map these semantic elements and relationships onto a linear string of phonologically encoded morphemes. These rules plus a dictionary must include both the regularities of the language and the word-specific idiosyncrasies of the language. Because of these idiosyncrasies the mapping is quite complex and rarely if ever, one to one. Relative to the problem of prefix analysis, these non-unique correspondences look something like this:

SEMANTICS	[reversal]		
LEXICON	un-	dis-	de-
	untie	disown	deactivate

Or seen from the perspective of form:

LEXICON			de-	
SEMANTICS	[reversal]	[remove from]	[get off of]	[derive from]
	deactivate	dethrone	detrain	deverbal

The same problem can also be seen in terms of the form and content of both the prefix and the root. The following chart shows the most viable possibilities. Under the PREFIX heading the pluses and minuses indicate whether the word at left has the form pre- and/or the content [before]. In the Root columns the signs indicate whether the characters remaining after the prefix form is removed constitute the form of a proper word and if this form has a meaning consonant with the content of the entire word.

	PREFIX		ROOT	
	Form	Content	Form	Content
Predetermine	+	+	+	+
Prevent	+	-	+	-
Predecessor	+	+	-	
Preach	+	-	-	
Foreknow*	-	+	+	+
Precede	+	+	+	-

It appears to me that only those words which would be marked with pluses in all columns should be considered as prefixed words for the purposes of the VIA program. In addition, if a dictionary of the type suggested by generative semanticists were available to the computer it would be possible to include words consisting of two bound elements as part of the productive processes of word formation. Such a procedure would allow bound morphemes which appear with minuses in the root column on the chart to be analyzed as compounds similiar to prefixed words. For example -ology must be marked minus under Form in the root

*This analysis of foreknow assumes of course that it has already been analyzed as prefix plus root.

column and -phone and -graph must be marked with a minus under Content since the free form does not have the same semantic content as the bound forms. However, these elements could be handled if proper entries for the bound forms were present in the dictionary. By providing such a dictionary which included bound elements as full semantic units and rules capable of replacing semantic subtrees with lexical units or combinations of units, the productivity of forms such as phone-, bio-, tele-, could probably be accounted for, although they do not qualify as prefixes within the definition given above.

Although the chart provides a clear objective for the discovery of prefixed words it does not provide any help in determining whether the root "has a meaning consonant with the content of the entire word." To make this kind of determination four kinds of information are necessary: 1) the meaning of the prefix, that is, the possible meanings of a particular prefix form, 2) the meaning of the word which remains when the prefix form is removed, 3) the rules for determining the semantic relationship of this particular prefix to the roots to which it is attachable, and 4) the meaning of the entire word. For example, to test the word prevent it is necessary to know what pre- means, what vent means, the semantic-syntatic relationship between roots and pre-, and the meaning of the complete form prevent. To be a prefixed word it must be the case that the sum of the information obtained in 1 through 3 is identical with that in 4. In other words to be a true prefixed word the meaning of the word must be predictable from the elements and their relationship.

The most difficult part of this analysis is that of formally stating the semantic-syntactic relationship which a prefix bears to

the root to which it is attached. In what follows I will show why this information is necessary and give some suggestions for analyzing a small sample of prefixes.

The following discussion should be understood within the general framework of transformational grammar and more particularly within the generative semantic framework of lexical insertion. This grammatical model, simply stated, contains a base component which generates semantic structures, representable as phrase structure trees, a transformational component which maps phrase structure trees onto phrase structure trees, a lexical component which contains the phonological, syntactic, and semantic information associated with lexical entries to be substituted in the phrase structure tree in the place of subtrees according to selectional restrictions. (For further discussion of generative semantics see Lakoff, 1969, Lakoff and Ross, forthcoming.)

I would like to suggest four reasons for deriving prefixes from underlying semantic-syntactic structures: 1) Some phrases are closely semantically related to prefixed words, 2) Structures are needed to account for ambiguity, 3) Prefixes alone can be questioned, commanded, or negated, and 4) Underlying structures provide an elegant way of expressing reversal.

I. Deriving prefixes from underlying phrase structures captures the generality that corresponding prefixed words and phrases are closely semantically related. Notice the following pairs of sentences:

1a. John opened the store again.

b. John reopened the store.

2a. Jane was not capable of crying.

b. Jane was incapable of crying.

3a. The linguistic student's grade before the examination was an A.

b. The linguistic student's preexamination grade was an A.

4a. The patient's condition after the operation was satisfactory.

b. The patient's postoperative condition was satisfactory.

5a. The child was too active.

b. The child was hyperactive.

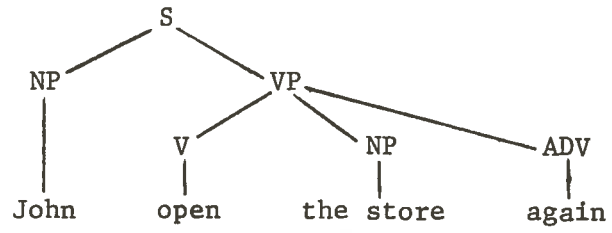
6a. The American colonies had no central government during the period before the revolution.

b. The American colonies had no central government during the prerevolutionary period.

The best way of showing the relationship of the a and b sentences is to derive them from the same underlying structures. This can be done by using transformations already needed for other structures and establishing lexical entries for prefixes with appropriate semantic descriptions and cooccurrence restrictions. The lexical rule would simply allow the insertion of a prefix when the structural tree matched that required by the prefix in question.

In the case of the sentences of 1 the structure would look something like 7. (I have ignored tense for simplicity.)

7.



The lexical entry for re- would contain the information that re- can be attached to a verb in structures containing an ADV node dominating the semantic unit [again], subject to selectional restrictions on the

verb (such as the verb must be transitive). If this option were taken the b sentence would result. If the lexical entry again were chosen for [again] the a sentence would result.

Specifically the rule for deriving reopen would have the power of a transformation in which the structural index would look like A:

A. X & V & NP & ADV & X

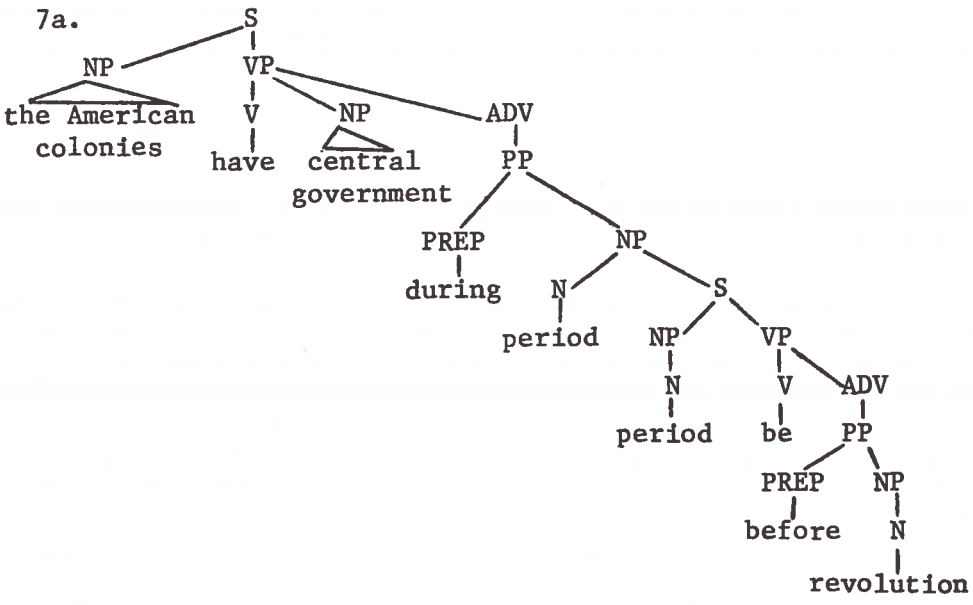
where & is a concatenation symbol, X is a variable which may be null or an indefinite string, (+ re-) is a feature notation on verbs which can take the re- prefix, and [again] is the semantic element under the ADV node. This rule would have to operate after the choice of the verb in order to satisfy the selectional restrictions. The structural change would look like B:

B. X & re-V & NP & X

where the dash (-) is a morpheme boundary. This rule would be part of the lexical entry for re- and assumes that the lexical entry for open contains the feature (+ re-).

Sentences 6a and 6b would have a structure roughly like 7a at some intermediate stage. (I have simplified the structure in all but the ADV nodes.)

7a.



The pre- rule relevant to this sentence would have the structural description of C:

C. X & N_1 & N_1 & V & PREP & N_2 & X
[before]

where the subscripts indicate identity of lexical or semantic items (I am not sure at this point which the tree would contain or if it might not contain both) and [before] refers to the semantic item which must appear under the PREP node. The structural change would be D:

D. X & pre-N_2 & N_1 & X

where pre- is a morpheme and the dash is a morpheme boundary.

There would also have to be a rule for adjectivization of revolution which would look something like E:

E. X & N_1 & N_2 & X

X & ADJ & N_2 & X

where the arrow indicates the structural change. There would also need to be a rule in the lexical entry for revolution which states something like F:

F. ADJ revolution

revolution-ary

where ADJ above revolution indicates that the word is dominated directly by the ADJ node.

These rules are specifically for these sentences and would doubtless require alteration to be applicable to a larger set of sentences. There are also a considerable number of problems involving suffixes which have a bearing on prefix analysis as is suggested by the revolution example above.

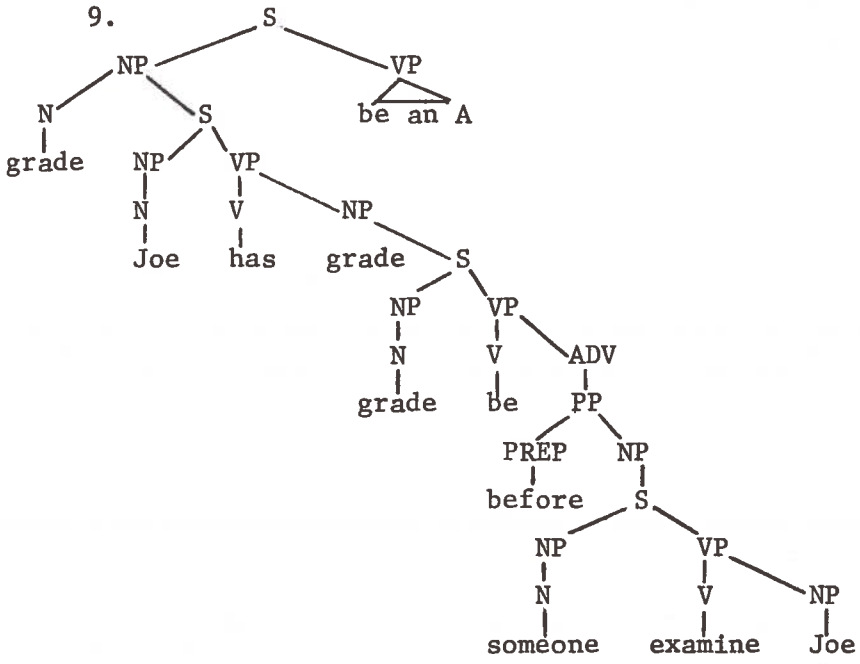
The other sentences could be analyzed in the same fashion, although I have not worked out the details of the lexical entries required nor the restrictions on the application of the entries. However, I see no real obstacle in the way of such an approach.

II. In addition to providing an explanation for the semantic relatedness of sentences such as those in 1-6, the proposed approach would also explain in a principled way ambiguities which occur with prefixed words. For example sentence 8 is at least two ways ambiguous.

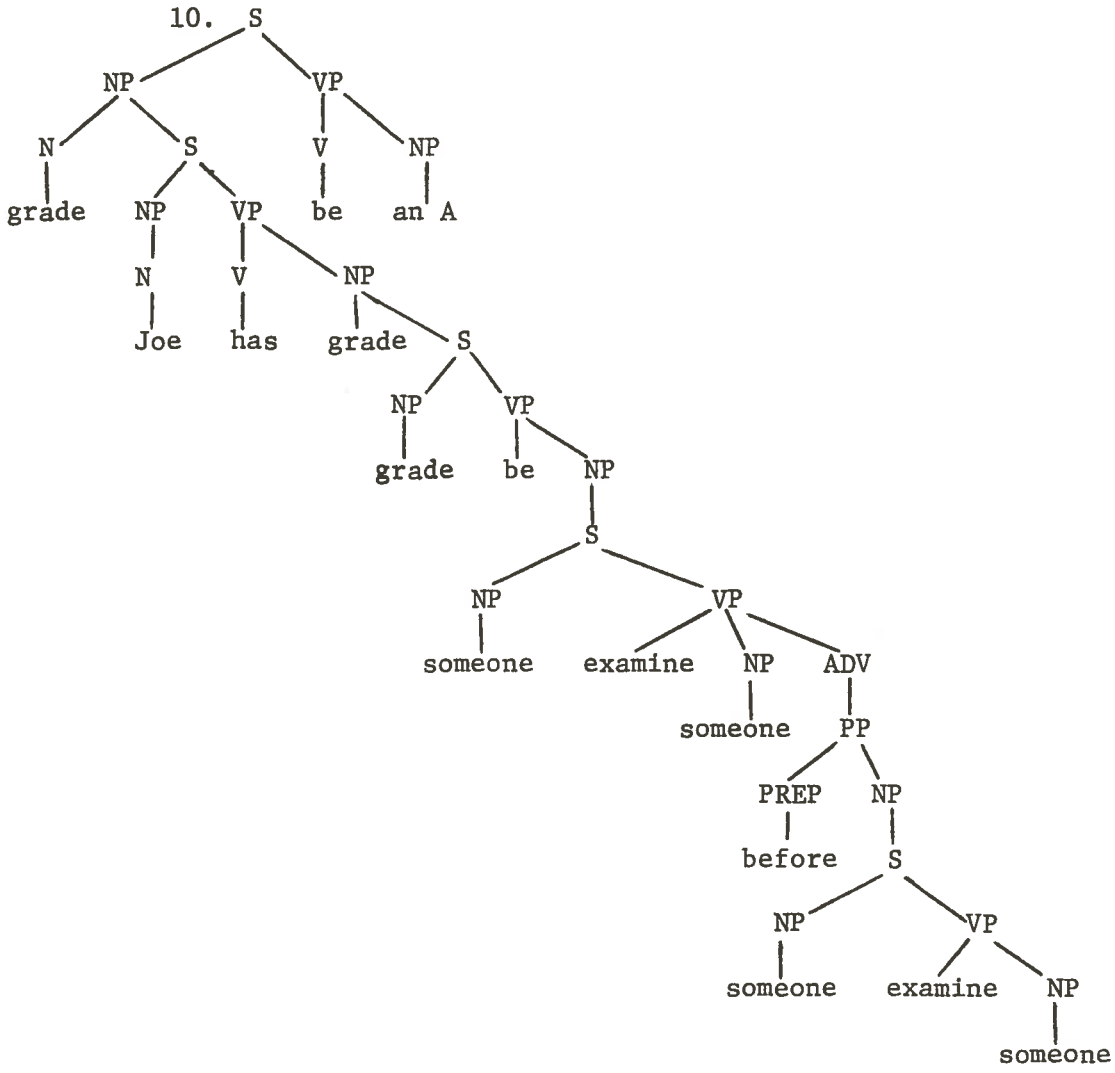
8. Joe's preexamination grade was an A.

In one reading the grade was earned before the examination; in the other the grade was for an examination which Joe took before another examination. These two semantic interpretations can be seen clearly if two separate underlying structures are posited for the readings. The first reading would have a structure something like 9.

9.



The underlying structure for the second reading would look something like 10.



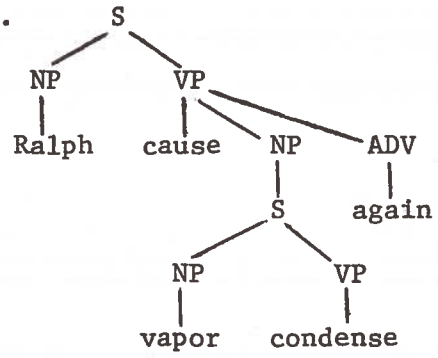
Whether noun modification and the nominalization of the S's containing [examine] are correct can be debated. However, the structures do show the advantage of treating prefixes as derived from underlying subtrees as a means of explaining the ambiguity.

Sentence 11 is another example of disambiguation which is possible through the proposed analysis.

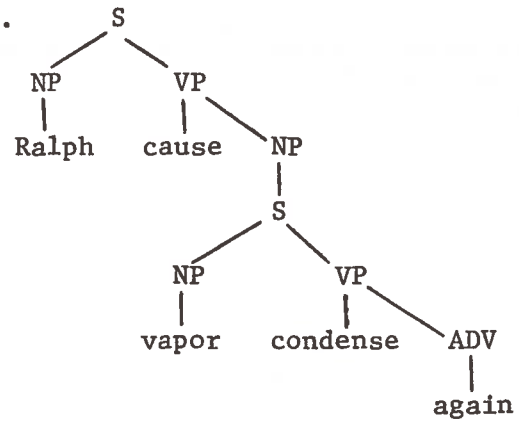
11. Ralph recondensed the vapor.

The sentence is ambiguous in that it is uncertain whether Ralph is condensing vapor which he condensed once before or whether Ralph is condensing vapor which he has never condensed before, but which has been condensed previously by someone else. Following Lakoff's (1965: IV) handling of causatives the ambiguity can be explained by the location of the adverb [again]. Notice structures 12 and 13.

12.



13.



These structures also underlie sentences 14 and 15 respectively.

14. Again Ralph condensed the vapor.

15. Ralph caused the vapor to condense again.

Sentence 16 is ambiguous in the same way as 11.

16. Ralph condensed the vapor again.

Notice that in order to indicate in the surface structure that it is the lower S which is the scope of the adverb in 13, it is necessary to include the verb of the higher S. If this analysis for re- is correct,

the causative verb plus the prefixed verb of the lower S should have the same reading as 15. Sentence 17 confirms this.

17. Ralph caused the vapor to recondense.

Thus sentences 15 and 17 are derived from 13, 14 is derived from 12, and 11 and 16 are ambiguous in that they may be derived from either 12 or 13.

III. Certain verbal prefixes require underlying phrasal structures to account properly for negation, questions and imperatives. Notice, for example, that in sentences 18, 19, and 20 it is the prefix which is being negated, questioned or commanded.

18a. Harold didn't resubmit his manuscript.

b. Sally didn't preregister.

c. Leo didn't co-edit the book.

19a. Did Harold resubmit his manuscript?

b. Did Sally preregister?

c. Did Leo co-edit the book?

20a. Resubmit your manuscript!

b. Preregister!

c. Co-edit the book!

In 18 it is not being denied that Harold submitted his manuscript, Sally registered, or Leo edited the book, but rather that Harold submitted the manuscript <u>again</u>, Sally registered <u>before registration</u>, and Leo edited the book <u>with someone else</u>. These facts cannot be explained unless the prefixes are derived from some underlying structure. The same is true of the questions and imperatives in 19 and 20.

If, however, the prefixes in question are derived from underlying adverbal constructions the facts can be explained in the same way as

adverbs with the same operators. Lakoff (1965:F-4) argues convincingly that it is necessary to consider at least some adverbs as higher VP's than the VP's with which they occur in surface structure. This analysis allows adverbs to be questioned, negated, and commanded in the way other verbs undergo these operations. To summarize his position he argues that in the sentences

21. Do you beat your wife enthusiastically?

22. Do you beat your wife in the yard?

the beating is assumed and only the adverb is questioned. Lakoff also argues that since it is necessary to question the adverb and Q is usually attached to the highest S then the adverb must be the predicate of a higher S than the verb which it "modifies" on the surface.

Lakoff (1970:154-7) also argues from sentences 23 and 24

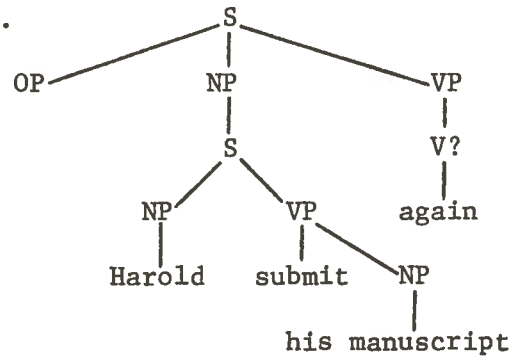
23. A Brink's truck was robbed outside of town, but it could never have happened within the city limits.

24. The train exploded near Mexico City, although the assassins had planned for it to happen at the border.

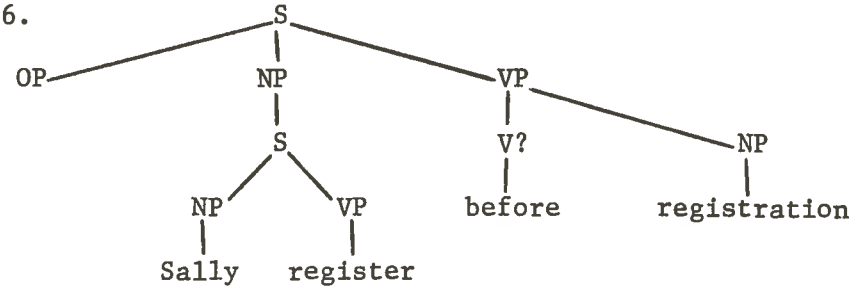
that the "it" in both sentences cannot refer back to "A Brink's truck was robbed outside of town" and "The train exploded outside Mexico City" respectively, but only "A Brink's truck was robbed" and "The train exploded." Thus, he concludes that the adverbal prepositional phrases "modify" the verbs "rob" and "explode" only in surface structure and are higher S's in the underlying structure.

Following this analysis, structures 25, 26, and 27 can be posited to underlie the sentences in 18 and 19, and with minor alterations the sentences in 20. The OP refers to NEG, Q, or IMP.

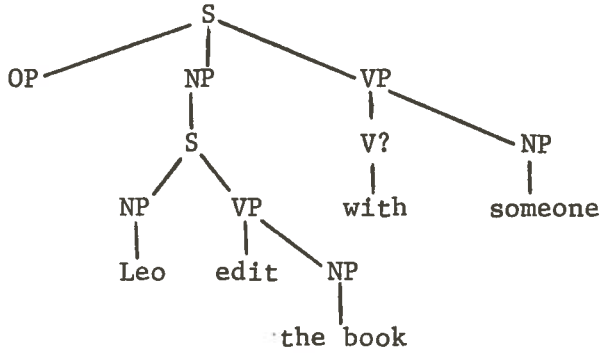
25.



26.



27.



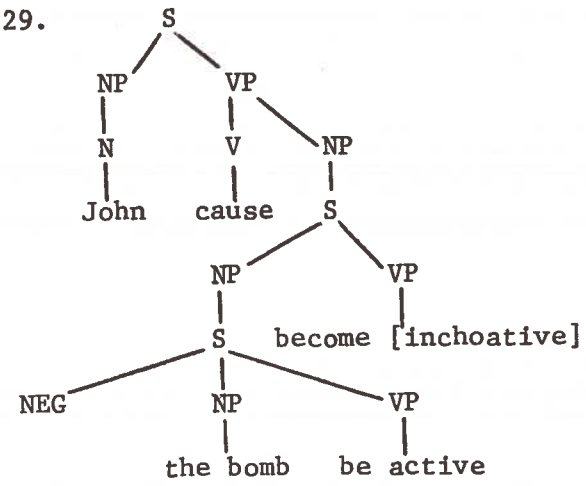
I am not sure how the nodes with question marks should be labeled, but it does appear that an analysis at least similar to the one given is necessary to account for the facts. I can see no other way to explain how only the prefix of a verb can be questioned, negated or commanded, unless the prefix is derived from some other structure.

IV. The final argument for deriving prefixes comes from Lakoff's dissertation (1965:IX 19ff.) in which he suggests as part of his argument for the pro-verbs [cause] and [inchoative] that the prefix de- can be derived from a negation of the lowest S of a causative-inchoative construction. His example is 28.

28. John deactivated the bomb.

He analyzes it as 29. (I have simplified part of the structure by leaving out his feature notation and by treating NEG as a node rather than a feature.)

29.



Without the NEG node this structure relates the following sentences which are closely semantically related.

30. John activated the bomb.

31. John caused the bomb to activate.

32. John caused the bomb to become active.

33. John caused the bomb to come to be active.

With the NEG node, sentences 30, 32, and 33 have proper parallels in 34, 35, and 36 respectively.

34. John deactivated the bomb.

35. John caused the bomb to become inactive.

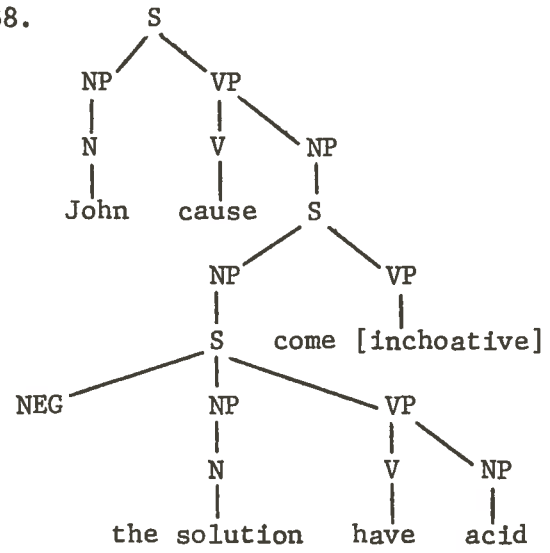
36. John caused the bomb to come to be not active.

This analysis provides a simple, elegant means for dealing with the semantic notion [reversal] and gives added evidence for deriving prefixes from more complex underlying structures.

Applying this approach to a large number of words in Webster's Third New International Dictionary with the prefix de- raises some questions, but does not force the abandonment of Lakoff's suggestion. For example, a number of words require that the lowest S have as its verb [have] instead of [be]. Sentence 37 can be analyzed as 38.

37. John deacidified the solution.

38.



The verb [have] here seems to have the meaning [contain] whereas other denominal verbs require meanings of possession, for example debeak.

Another problem with this analysis is in determining the form of the lowest S. Does decompress come from [to cause to come not to have compression] or from [to cause to come not to be compressed]? Does decongest come from [to cause to come not to have congestion] or from [to cause to come not to be congested]? The solution lies in the proper understanding of the relationship between the surface verbs be and have in their underlying semantic structure.

In summary I think that prefixes can best be dealt with by using semantic information, and that this semantic information is best handled by generative semantics. Further research into prefixes within

this framework should provide the principles necessary for making the CLUD lists of PREFIX and may modify the theory of generative semantics.

The research should be of practical help in compiling the CLUD lists by providing a structural frame into which the word to be tested for a prefix is inserted. If the resulting meaning corresponds to the dictionary meaning then it should be considered to be prefixed. For example words with re- can be placed in a structural frame something like 39.

39. John _____ something again.

If the analysis is correct the meaning should be very similar to 40.

40. John re-_____ something.

In its most usual reading react would not pass such a test, since it does not mean to [act again]. I hope to be able to devise better and more reliable tests for this and other prefixes.

REFERENCES

Lakoff, George. 1965. On the Nature of Syntatic Irregularity. Computational Laboratory of Harvard University, Mathematical Linguistics and Automatic Translation, Report No. NSF 16 to National Science Foundation, Anthony G. Oettinger, Principal Investigator, Cambridge.

_____. 1970. Pronominalization, Negation, and the Analysis of Adverbs. In Jacobs, Roderick and Rosenbaum, Peter. Readings in English Transformational Grammar. Waltham, Massachusetts: Ginn and Company.

_____ and John Robert Ross. Forthcoming. Abstract Syntax.

SUPPLEMENTAL BIBLIOGRAPHY ON PREFIXING

(See original bibliography in Research Report for 1 March 1969)

Annear, Sandra & Elliot, Dale E. "Some Problems of Derivational Morphology," Working Papers in Linguistics, Report #1, The Ohio State University, (1967), 110-15.

Bankevitch, L. English Word-building. Leningrad, 1961.

Brown, Rowland Wilbur. Materials for Word-study, a Manual of Roots, Prefixes, Suffixes and Derivatives in the English Language. New Haven, 1927.

Chapin, Paul. On the Syntax of Word-derivation in English. Information System Language Studies Number Sixteen. Mitre Corporation. Bedford, Massachusetts, 1967. ERIC No. ED 019 648.

_____. A Procedure for Morphological Analysis. ERIC No. ED 022 148.

Festschrift Hans Marchand. The Hague. 1968.

Givon, Talmy. Transformations of Ellipsis, Sense Development and Rules of Lexical Derivation. Systems Development Corporation, Santa Monica, California, 1967.

Heller, John L. & Swanson, Donald C. Elements of Technical Terminology. Champaign, Illinois, 1962.

Hietsch, Otto. "Moderne Englische Wortbildungselemente." Anglistische Studien. Festschrift zum 70. Geburtstag von Professor Wild. Wiener Beitrage zur Englischen Philogie 66. Wien/Stuttgart, 1958, 81-101.

Lakoff, George. On the Nature of Syntactic Irregularity. Computational Laboratory of Harvard University, Mathematical Linguistics and Automatic Translation, Report No. NSF 16 to the National Science Foundation, Anthony G. Oettinger, Principal Investigator, Cambridge, 1965.

Klima, E. S. "Negation in English." In Fodor, Jerry & Katz, Jerrold. The Structure of Language. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1964.

Koziol, Herbert. Handbuch der Englischen Wortbildungslehre. Heidelberg, 1937.

_____. "Zur Aufnahme von Wortneubildungen im Englischen," Orbis. 1955, 4:452-58.

_____. "Zur Wortbildung im Amerikanischen Englisch." Anglistiche Studien. Festschrift zum 70. Geburtstag von Professor Wild. Wiener Beitrage zur Englischen Philogie 66. Wien/Stuttgart, 1958, 127-38.

Lees, Robert. A Grammar of English Nominalizations. International Journal of American Linguistics. 26:3, Part 2, 1960.

Marchand, Hans. "The Negative Verbal Prefixes in English." Melanges de Linguistique et de Philologie. 1959, 267-76.

_____. "Notes on English Prefixation." Neophilologus. Groningen, 1954, 55:294-304.

Revard, Carter C. Affixal Derivation, Zero Derivation, and "Semantic Transformations." Systems Development Corporation, Santa Monica, California. Technical Memorandum 3835. 1967.

Schultink, H. "Productiviteit als Morfologisch Problem. Forum der Letteren. Leiden, 2:110-25, 1961.

Soboleva, P. A. "O transformacionnom analize slovoobrazovatel'nych otnosenij." (Transformational Analysis of Word-building Relations [English Summary]). In Transformacionnj metod v strukturnoj lingvistike. Moskva:Izd. "Nauka" 1964, 114-41.

Wilhelmsen, L. J. "On the Verbal Prefixes for and fore in English." Avhandlinger utgitt av Det Norske. Videnskaps-Akademi i Oslo. No. 2, 1938.

Zimmer, K. Affixal Negation in English and other languages. Monograph No. 5, Supplement to Word 20, 1964.

II. Program Documentation

A. 1. INTRODUCTION

This section of the report contains documentation for the FORTRAN version of the list-structure VIA as it is currently implemented on the Honeywell 635. In addition to the documentation provided in this section, program listings of INDEX and SUFFIX are given in the Appendix (Section IV of this report).

A. 2. IMPLEMENTATION CONSIDERATIONS

by Frank Joyce

The FORTRAN IV VIA package closely parallels the PL/1 version in terms of function, though it does differ somewhat in execution strategy. The PL/1 version was developed for use at IBM S/360 installations. Unfortunately, at present PL/1 is not generally available at non-IBM installations. Therefore, when work on this project was shifted to the University of Kansas, which has a Honeywell 635 computer, it was necessary to reprogram the system. Although some consideration was given to languages such as COBOL or SNOBOL which have nice character manipulation features, it was decided that the reprogramming should be done in FORTRAN IV due to its wide availability and the greater standardization of compilers among manufacturers. During the reprogramming process, every effort was made to provide a system which would be "machine independent". However, at certain points it was necessary to make some compromises due to FORTRAN's weakness as a text processing language, and this section will point out some of the potential problems in implementing the system at a different location.

CHARACTER MANIPULATION IN FORTRAN

FORTRAN IV is a mathematical language best suited for scientific and engineering applications, thus character manipulation is rather inefficient and this version should only be used if PL/1 is unavailable at the user's computer installation.

The Honeywell 635 computer has 36-bit words and represents alphameric characters in the BCD code, which employs six bits per character. Thus, up to six characters could be stored in each computer

word. However, in order to manipulate individual characters, it is necessary that only one character be stored in each computer word. This character is stored in the leftmost character position, just as if it had been read in under the FORTRAN 1A1 format. In programs such as SUFFIX which require extensive tables to be stored in core memory, such inefficient use of storage capacity would be prohibitively expensive. Thus, these tables are "packed"; that is, six characters are stored in each computer word. Therefore, the user must provide two subroutines which "pack" and "unpack" strings of characters. These two subroutines, COMPCT and BRKUP, are discussed below.

A complication was encountered when trying to decide if one character was higher than another in alphabetical sequence. Recall that the character is stored in the leftmost character position of the computer word. Note also that FORTRAN compares numbers, not alphameric characters. This last fact does not present a serious problem in general since the computer word containing the character can be treated as though it held an integer number. In general, if one of two characters is higher in collating sequence, then the word containing its code will represent a larger number than the number contained in the second computer word. However, if the leftmost bit of the computer word is treated as a "sign bit" in arithmetic operations, as is the case with many computers, then it is possible that the larger character has turned on this bit, thus appearing to be a negative integer and hence smaller than the character that is "actually" smaller. Therefore, the third user supplied subroutine,

LCOMP, does a "logical" comparison of the characters rather than an "arithmetic" comparison.

USER SUPPLIED ROUTINES

The user must supply three subroutines which carry out tasks that are dependent on the architecture of the user's computer. While it is possible that these routines could be written in FORTRAN, it is recommended that they be written in assembly language.

1. COMPCT. This subroutine takes the characters which are stored in a dimensioned array, one character per computer word, and creates a string in which the characters are packed at maximum density (on the Honeywell 635, six characters per computer word).

Calling sequence: CALL COMPCT(NDEST,NSOURC,NOCHAR) where

NDEST is the first word of the destination (packed) string,

NSOURC is the first word of the source (unpacked) string, and

NOCHAR contains the number of characters to be transferred.

EXAMPLE: (In this example assume the following dimension statement:

DIMENSION NDEST(3),NSOURC(18))

```
CALL COMPCT(NDEST(1),NSOURC(2),15)
```

String NSOURC contains 18 characters, one per computer word. Fifteen of these characters, beginning with the second character, will be packed into the string NDEST, beginning in the leftmost character position of the first word of the array NDEST. String NSOURC will remain unaltered.

2. BRKUP. This subroutine takes the characters found in a packed string and creates a duplicate string with one character stored

per computer word.

```
CALL BRKUP(NDEST(3),NSOURC(2),2)
```

The last 12 characters of string NSOURC will be placed in string NDEST beginning with the third word of array NDEST. String NSOURC will remain unaltered.

3. LCOMP. This is a function subprogram which does a logical comparison of two computer words.

How to use LCOMP: JTEST = LCOMP(WORD1,WORD2)

If WORD1 is higher than WORD2 in the collating sequence, then JTEST will equal 1. If WORD1 is lower in the collating sequence than WORD2, then JTEST will equal -1. Finally, if the contents of WORD1 and WORD2 are the same, then JTEST will equal 0.

EXAMPLE:

```
     DO 10 J=1, 10

     JTEST = LCOMP(WORD1(J),WORD2(J))
     IF(JTEST) 20, 10, 40
10   CONTINUE
```

In this example, assume that WORD1 and WORD2 are strings of characters, one character per computer word. If the string in WORD1 is the higher of the two, with respect to the first ten characters, then control will transfer to statement 40. If the first ten characters are the same, then control will go to the statement immediately following statement 10. Finally, if the string in WORD2 is the lower of the two strings, then control will transfer to statement 20.

FILES

Records written to magnetic tape or disc are written unformatted. That is, they are written out (and later read back in) in binary with no alteration to the contents of the computer words being written out (or read in). The READ and WRITE statements do not refer to any FORMAT statements. If this is not permitted in the version of FORTRAN implemented at the users location, it will be necessary to add the formats.

EXAMPLE:

```
WRITE (10) ISUFF1,ISUFFL
```

A. 3. INDEX PROGRAM USER GUIDE

by Thomas Kosakowski

INDEX is a multifaceted indexing program that takes as input 80 character punched cards or card images on some other medium such as magnetic tape. Provisions are made for input from a variety of natural language sources. General processing modes exist for prose, poetry, both verse and prose plays, and transcription of spoken text: there is a special processing mode for long poems, such as Milton's Paradise Lost, which may be divided into books, or cantos, or something analogous, as well as lines--because Paradise Lost is the one such case with which INDEX has dealt, this mode is called MILT. It is assumed that text will be found in columns 1-71 of each card image. A hyphen for the continuation of a word should be placed in column 72. Page numbers may be placed in columns 73-75, and if the user wishes to put sequence numbers on his data cards - in case the deck is dropped - he should do so in columns 76-80. If page numbers are used they will override volumne volumne count in MILT and stanza count in POET. All elements that are to be recognized by the computer as independent must be separated by spaces. This convention implies, in addition to words, all punctuation marks are to be separated by blanks. For example, the ending of the last sentence would be punched " separated by blanks . " Textual division and paragraphs in PROSE or lines in POET are indicated by other conventional markers. These will be discussed below.

Input for all processing modes consists of two files called SYSIN and INFILE. The SYSIN file (FORTRAN logical unit 5) should

have a header record and a title record. The format of the header record is:

PROC=PROS,	PRINT=YES,	SEQ=YES,	STAGE=YES,	DELIM=character
PROC=MILT,	PRINT=NO,	SEQ=NO,	STAGE=NO,	DELIM=$
PROC=SPOK,				
PROC=POET,				
PROC=PLAY,				
PROC=VPLA,				

Braces indicate alternatives; one must be chosen. Brackets indicate optional alternatives; the default is underlined. The title record may contain any information. It will be printed as a page heading if the PRINT=YES option is chosen. The INFILE records consist of text material in the formats described below. The DELIM=character option is used to specify the control character for end of paragraph (default $$$) and resetting of index numbers (default $PARAGRAPH= xxx, etc.). The specified character will replace the dollar sign wherever it is used in the following instructions.

Output appears on two files: SYSPRT (Fortran logical unit 6) for printed output, and OUTFIL (Fortran logical unit 21) for text material.

Output for all processing modes consists of 36 binary computer word logical records containing indexing information and the word or punctuation mark itself. A printed listing of this output may be obtained by putting 'PRINT=YES' as the PARM value in the header record. If no printed listing is desired, use 'PRINT=NO'. If the processing mode is PLAY, stage directions can be included in the data set by using the statement, 'STAGE=YES' in this same statement. 'STAGE=NO' will result in the omissions of these portions of input

text. See below for how to indicate stage directions.

PROS is the processing mode for prose text. It will be used as the main example in this discussion. In addition to being blank delimited, as discussed above, the text should have the following markers for major textual division:

Paragraph: end of paragraph is indicated by a double period (..) instead of the single period for the last sentence. Paragraphs ending with some other mark of punctuation such as ! or ?, should include the punctuation mark, followed by a space and a double ".." .

Chapter: end of chapter is indicated by $$$.

Volume: end of volume is indicated by $$$$.

Page: (Optional) place appropriate number in columns 73-75.

Sequence: (Optional) place appropriate number in columns 76-80.

The index numbers for volume, chapter, and page as well as the sequence number may (at any point of data) be set to begin with a user-specified value by placing in the data set a card with a '$', the appropriate index level and the value to which this counter is to be set. For example, if the user wishes to begin his paragraph counter with paragraph five then he should insert a card with '$PARAGRAPH 5' on it. Similar updating procedures apply for most processing modes.

The Output record will have the following format:

1	1	1	1	1	1	1	1	1	1	8	18	(length in computer words)
LINEAR # IN TEXT	VOLUME NUMBER	CHAPTER NUMBER	PARAGRAPH NUMBER	SENTENCE NUMBER	WORD IN SENTENCE	PAGE NUMBER	IDIOM FLAG	PREFIX LENGTH	WORD LENGTH	PREFIX	WORD	

One record which contains zero in the linear # field and special characters in the word field is written by the index program at termination. When the file is sorted on the word field this record will always sort to the end of the file. The presence of the zero linear # field is used to initiate end-of-file processing by following programs.

The prefix field is used by the PREFIX program if prefix information is desired. Otherwise the field may be used by the researcher for categorizing information or for whatever use he may have in subsequent programs. The idiom flag will be set to 1 if the WORD represents a group of words used as a single, idiomatic expression; otherwise, it will be zero (0). This feature is not operative at present. The length field contains the number of characters in the WORD. Thus, an actual output record might look like this:

34	1	1	1	3	4	12	0	1	2	b	b	b	b	b	b	b	b

In this example, the word <u>at</u> appears as the 34th word in the text; and it is the fourth word in the third sentence of paragraph one, volume one, of the text. Also, it is found on page twelve; it is not

an idiom, and it is of length two.

For storage of his output, the user should provide a data set on some medium such as tape or disk with Fortran logical unit 21.

MILT is the special processing mode for Paradise Lost and analogous texts. Input should be blank delimited, one line per card image. If a line is continued from one card to the next, indicate this with an "@" in column 72. Volume numbers should be placed in columns 73-75, and sequence numbers in columns 76-80, if used.

Paragraphs (if used) are indicated by double periods (..).

Books or chapters are indicated by $$$.

Volume is as indicated in columns 73-75.

No update facilities exist for this processing mode. Printout and sequence checking are indicated as in PROS. Output is identical to that from PROS except that volume is usually one, line number replaces sentences, and word in sentence becomes word in line. Again, the user should supply an output data set for logical output records.

PLAY is the processing mode for prose plays. Printout, and sequence checking are indicated as before, If stage directions are to be included in the data set, use 'STAGE=YES', otherwise 'STAGE=NO'. Regardless of whether or not they are to be included, they should be punched with an asterisk preceding each word; for example, (*lights *dim *slowly *.)

Index information generated is act, scene, paragraph, sentence word in sentence.

Paragraph is indicated by double periods (..).

Scene is indicated by $$$.

Act is indicated by $$$$.

Act and Scene can be reset by using $ACT ### or $SCENE ###.

Output records are as before except that act replaces chapter and scene replaces paragraph. The user should provide an output data set.

VPLA should be used for plays that are written in verse. Printout and sequence checking should be indicated as described on pages 2 and 3. If stage directions are to be included in the data set then place 'STAGE=YES' on the hdr card. As was the case for PLAY, stage directions should be punched with an asterisk (*) preceding each such word regardless of whether or not they are to be included in the output data set. Data should be punched with one line of text per card.

Index information generated is volume, act, scene, line, and word in line. Scene and act are indicated as for PLAY. Act and Scene numbers can be reset as in PLAY.

As before, the user should provide an output data set.

POET is the processing mode for poetry. Text should be punched one line of poetry per card. If a line must be continued from one card to another, place an 'at' sign (@) in column 72. If the poetry is stanzaic, then the user must place a number which indicates the stanza in columns 73-75. Sequencing numbers, as before, should be placed in columns 76-80. Indexing information generated will be stanza, line, and word in line. No updating facilities exist for this mode. Volume, chapter, and page will all appear with the value one (1) in the output data set. As before, the user should provide an output data set.

SPOK is a processing mode for transcription of oral discourse.

Indexing information generated will be for series, session, speaker, sentence, and word in sentence. Series, session, and speaker counters can be reset by inserting a card on which '$', level, and the desired number have been punched. For example, '$SPEAKER 3' would set the speaker counter at an initial value of three. Within the data itself the following conventions hold:

A change in SPEAKER is indicated by double periods (..).

A change in SESSION is indicated by $$$.

A change in SERIES is indicated by $$$$.

As before, page numbers and sequence number should be placed in columns 73-75 and 76-80 respectively.

PROGRAMMER'S GUIDE TO INDEX PROGRAM

The following describes the functions of the various modules of the INDEX program.

Module: CLOCK1 (IDAY, IMON, IYR)

Description: This module returns the current date in A2 format.

Output arguments:
IDAY - day
IMON - month
IYR - last 2 digits of the current year

Common variables used: IDAY,IMON,IYR

Module: CONVRT (WORD, WRDPTR, NUM)

Description: This module converts a substring in the text string in A1 format to an integer variable.

Input arguments:
WORD - array containing the text substring
WRDPTR - integer subscript pointing to the first blank to the right of the text substring

Output arguments:
NUM - integer variable

Common variables used: none

Modules called: none

Module: ENDPAG

Description: This module is called to print a page heading before an error message is printed.

Input arguments: none

Output arguments: none

Common variables used: TITLE, IMON, IDAY, IYR, PAGE, LINES, FIRSTF.

Modules called: none

Module: FLGEOF (INFILE,EOFLAG)

Description: This module checks an input file for end-of-file condition and sets a switch accordingly.

Input argument: INFILE - integer representing file to be checked for EOF condition

Output argument: EOFLAG - 0 normal read
1 EOF encountered

Common variables used: none

Modules called: none

Module: GETWRD

Description: This module scans text records and picks out words delimited by blanks. It also rejoins word segments that are continued from one text record to another.

Input arguments: none

Output arguments: none

Common variables used: WORDS, READF, LIMCOL, CONTIN, CHAR.

Modules called: READIT

Module: INITCM

Description: This module initializes some common variables as housekeeping prior to the processing of parameter cards.

Input arguments: none

Output arguments: none

Common variables used: SYSIN, SYSPRT, INFILE, OUTFIL, PAGE, LINES, LINEPG, PRINTF, SEQFLA, SEQOLD, STAGEF, FIRSTF, READF, TYPEPR, WRDPTR, CARDCN, OUTCNT.

Modules called: none

Module: MAIN

Description: This module is executed first for all types of text

processing. It reads the processing parameter and title cards. The parameter card is scanned and various logic switches are set. Control is then given to the indicated processing module: PROS, MILT, PLAY POET, or SPOK. Control is returned to this module at end-of-file of the text material. Final statistics are printed, files are closed, and the INDEX program terminated.

Input arguments: none

Output arguments: none

Common variables used: WORDS, EOFLAG, IDAY, IMON, IYR, INFILE, SYSIN, TITLE, TYPEPR, PRINTF, SEQFLA, STAGEF, FIRSTF, WORDN, SYSPRT, WRDPTR, OUTFIL, INFILE.

Modules called: FLGEOF, CLOCK1, PRINT, PROS, POET, PLAY, MILT, SPOK, ENDPAG, WRTWRD, INITCM, EXIT.

Module: MILT

Description: This module processes text in a manner similar to the module PROS when the Milton mode of processing is indicated.

Input arguments: none

Output arguments: none

Common variables used: LIMCOL, WORDN, CNTRS, WRDPTR, PAGEN, READF, WORD, EOFLAG, WORDS.

Modules called: GETWRD, WRTWRD.

Module: OFLO

Description: This module is called to print page headings when the "PRINT=YES" option is chosen on the processing parameter card.

Input arguments: none

Output arguments: none

Common variables used: TITLE, IMON, IDAY, IYR, PAGE, TYPEPR, SYSPRT, LINES.

Modules called: EXIT

Module: PLAY

Description: This module processes text in a manner similar to the module PROS when the "play" or "verse play" mode of processing is indicated on the parameter card.

Input arguments: none

Output arguments: none

Common variables used: WORDN, CNTRS, TYPEPR, LIMCOL, WRDPTR, WORD, STAGEF, READF, EOFLAG.

Modules called: GETWRD, CONVRT, WRTWRD.

Module: POET

Description: This module processes text in a manner similar to the module PROS when the "poetry" mode of processing is indicated on the parameter card.

Input arguments: none

Output arguments: none

Common variables used: LIMCOL, CNTRS, WRDPTR, PAGEN, READF, WORD, WORDS, EOFLAG.

Modules called: GETWRD, WRTWRD.

Module: PRINT (ALPHA)

Description: This module prints the options chosen on the processing parameter card or the defaults taken for options not specified.

Input arguments:
ALPHA - type of processing

Output arguments: none

Common variables used: FIRSTF, SYSPRT, LINES, PRINTF, SEQFLA

Modules called: OFLO, ENDPAG.

Module: PROS

Description: This module processes text in the "prose" mode including reading text words, updating statistics counters, resetting statistics counters upon user request, and writing text word records.

Input arguments: none

Output arguments: none

Common variables used: WORDN, CNTRS, LIMCOL, WORD, WORDS, WRDPTR, EOFLAG.

Modules called: GETWRD, CONVRT, WRTWRD.

Module: READIT

Description: This module reads a physical text record from the input device and performs sequence checking if requested by the user.

Input arguments: none

Output arguments: none

Common variables used: WORDS, INFILE, CHAR, CONTIN, PAGE, SEQ, SEQOLD, SEQFLA, CARDCN, EOFLAG, SYSPRT.

Modules called: none

Module: SPOK

Description: This module processes text in a manner similar to the module PROS when the "spoken" text mode of processing is indicated.

Input arguments: none

Output arguments: none

Common variables used: WORDN, CNTRS, LIMCOL, WORD, WORDS, WRDPTR, EOFLAG.

Modules called: GETWRD, CONVRT, WRTWRD.

Module: SYMBOL

Description: This is a named common block used to load alphabetic

characters into the common area.

Input arguments: none

Output arguments: none

Common variables used: WORDS

Modules called: none

Module: WRTWRD

Description: This module is called to output a text word record after all processing has taken place.

Input arguments: none

Output arguments: none

Common variables used: OUTCNT, WRDPTR, WORD, BLANKS, WORDL, WORDN, CNTRS, PAGEN, PRINTF, SYSPRT, LINES, LINEPG.

Modules called: OFLO

Use of common variables:

Common variable	Function
SYSPRT	logical unit number for printed output
SYSIN	logical unit number for reading processing parameter and title cards
INFILE	logical unit number for reading text
OUTFIL	logical unit number for output text records
PAGE	page counter for printed output
LINES	line counter for printed output
TYPEPR	type of processing (PROS, POET, etc.)
SEQ	sequence number of current input text record
SEQOLD	sequence number of last input text record
LIMCOL	column number of last text character in text record
CHAR	array for text field of input text record
TITLE	array holding title card for page headings
PAGEN	page number field from input text record
CONTIN	continuation character from input text record
WRDPTR	subscript pointing to next character to process in WORD array
CARDCN	count of input text records
CNTRS	counters for volume, chapter, paragraph, sentence, and word-in-sentence for PROS mode. Similar for for other modes.
WORDN	linear text word number in output record
WORDL	number of characters in word in output text record
WORD	array containing word currently being processed
OUTCNT	count of output text records
IDAY	day of current date
IMON	month of current date
IYR	year of current date
LINEPG	lines per page for printed output
PRINTF	.TRUE. for "PRINT=YES" option
SEQFLA	.TRUE. for "SEQ=YES" option
STAGEF	.TRUE. for "STAGE=YES" option
FIRSTF	.TRUE. when a page heading should be printed
READF	.TRUE. for first text word of each physical text input record
EOFLAG	set to 1 when EOF encountered on INFILE

Machine Dependency

The INDEX program is written so that most modules are machine independent. The modules that may need to be changed if the program is run on a machine other than a GE600 are CLOCK1, FLGEOF, READIT, and SYMBOL. The changes are:

CLOCK1 - substitute another subprogram to provide the date.

FLGEOF - substitute another subprogram to handle end-of-file checking or implement an end-of-file data record and suitable checking in the READIT module.

READIT - if the Fortran of the new machine will not handle special characters such as the question mark (?) the READ statements must be changed. Substitute a call to an assembly language subprogram.

SYMBOL - change the DATA statements to reflect the internal character coding of the new machine. Change the DATA statements to reflect the syntax requirements of the new FORTRAN compiler.

The INDEX program requires approximately 14,000 words of memory on a GE600 (36 bit word length).

A. 4. PREFIX

by Frank Joyce

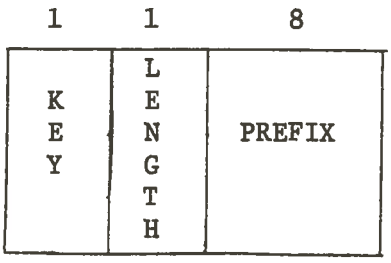
PREFIX is a program which identifies legal English prefixes, using a table look-up procedure. The program accepts as input the file created by program INDEX, sorted in alphabetical order. The output file should be resorted into alphabetical order on the word field before being used as input to program SUFFIX. Two linguistic restrictions have been placed on prefixes:

1. The prefix must be a bound morpheme.
2. A word is considered to have a prefix only if the remainder of the word, without the prefix, is independent, i.e., not a bound morpheme.

After determining that a word has a legitimate English prefix, the program moves the prefix to the prefix field of the input record and shifts the remainder of text word to the left edge of the word field.

Besides the text file, PREFIX accepts two other input files, a file of prefixes and a file of "clud words". The clud words indicate when a word with initial characters matching a perfix is a legitimate prefixed word, rather than just beginning with the same letters. For some prefixes it is easier to compile a list of words beginning with the same letters, but which are not legitimately prefixed. In this case the prefix is tagged with a 0 in the key field (see record format below) to indicate that the clud words form an exclusion list. On the other hand, for other prefixes it is easier to compile a list of words for which the prefix is legitimate. In these cases the prefix has a key of 1 indicating that the clud words form an inclusion list.

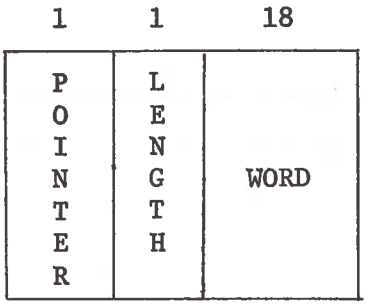
The prefix records have three fields; the key field; a field containing the length of the prefix; and the field containing the prefix itself. The format follows (with lengths in computer words):



The prefix itself is stored with one character per computer word. That character is stored in the leftmost character position of the computer word, as though it had been read in under a 1A1 format specification. The program does not read in the entire prefix file at once. Rather, all the prefixes beginning with the same letter are read in. When the text word no longer begins with that letter, a new batch of prefixes is read in.

The clud word file is sorted in alphabetical order. The clud word records also have three fields. The first field points to the prefix that corresponds to the clud word. For example, "ABED" is a clud word corresponding to the prefix "A". Since "A" is the first prefix beginning with an "A", the pointer field would contain "1". "ABBERANT" is a clud word for the prefix "AB" and "AB" is the second prefix beginning with "A", thus the pointer field for this word would contain "2". The second field is a length field containing the number of characters in the clud word. The third field contains the clud word itself. The text word is compared only to the letters in the clud word field and, therefore, the clud word "ATYPI" will match with

both "ATYPICAL" and "ATYPICALLY". If it is desired that the text word match the clud word exactly, then the clud word must have the blank character indicated as part of the clud word. Thus, "APPEND" will match with the text word "APPEND" but will not match with the text word "APPENDIX". The record format follows, with length in computer words.



The clud word field is treated just as explained above for the prefix field.

The alphabetized text file is processed with a single pass against the clud word file, which is also alphabetized. A scan of the clud list reveals whether a text word is on the list or not. The main processing procedure then branches into two simple nested conditional logic paths. If the word is found, then a test of the associated key is made, making use of the pointer to the prefix record. If the key is a one, indicating an inclusion list, a test for prefix match is made. If all three tests are successful, then we can say that the word has a legitimate English prefix. On the other hand, if the text word is not found in the clud list, a quick search of the prefixes beginning with the same letter of the alphabet as the text word is made. Those, and only those, prefixes with a key of zero are tested against the initial letters of the word. If a match is found, we can conclude that the word has a prefix.

FILES

1. Alphabetized clud word file--FORTRAN logical unit 10.
2. Alphabetized prefix file--FORTRAN logical unit 12.
3. Alphabetized text file from program INDEX--Logical unit 15.
4. Output file which should be re-alphabetized before use by program SUFFIX--FORTRAN logical unit 20.

A. 5. SUFFIX USER'S MANUAL

by Frank Joyce

SUFFIX

SUFFIX is a program that groups words having the same root. It does not "strip" the various suffixes from such words but leaves the words as they appear in the text, indicating that they have the same root by assigning to each word-token in a root group the same number, called a matchcount. Thus, for example, all tokens, or records, for complete, completely, and completing would have the same matchcount, for example 536. This number has only relative meaning. The next root-group, in alphabetical sequence, would have the matchcount 537. By this number one can determine words that have the same root form, but which differ in ending, thereby facilitating analyses such as thematic and semantic studies which may not involve syntax.

Input is assumed to be logical records of 36 computer words, sorted in alphabetical order on the text word field, from either PREFIX or INDEX. Output will be 29-word records, identical to input records except for the addition of the matchcount number, the number of word tokens for each match group and the number of tokens for each unique word type. This record has the following format:

(length in words)

1	1	1	1	1	1	1	1	1	1	1	1	1	8	18
LINEAR #	VOLUME	CHAPTER	PARAGRAPH	SENTENCE	WORD IN SEN	PAGE #	IDIOM FLAG	MATCH COUNT	MATCH FREQ	TOKEN FREQ	PREFIX LENG	WORD LENG	PREFIX	WORD

If the user wishes to eliminate function words from this data set before calculating match counts, he may do so by using "FUNCT=NO" on the first parameter card (see below). Similarly, punctuation marks may be deleted by stating "PUNCT=NO". Otherwise they will be included, with all function words grouped together with a matchcount of 99999 and all punctuation marks with 99998. Unless "PRINT NO" is included on the second parameter card, a printed copy of the output will be produced. An "Audit and Error Listing" report is also produced (see "Audit and Error Messages" below). This report will include a listing of function words and punctuation marks.

PARAMETER CARDS

1st Card (MUST begin in column 1)

```
FUNCT=YES ,PUNCT= YES
      NO          NO
```

There must not be any embeded blanks between any symbols in the parameter string.

EX. FUNCT=NO,PUNCT=YES

2nd Card

```
col 1          10     15

    PRINT= YES nnnnn nnnnn
           NO
```

The two five-digit integer fields must be right justified. The first field, which is in columns 10-14, specifies the number of lines per page for the output report. The second field, columns 15-19, specifies the number of lines per page for the "Audit and Error Listing" report. If either field is not specified, the default value will be

60 lines.

EX. PRINT=YES0007500060

3rd Card (Format control. MUST begin in column 1 with no embeded blanks. The specified order must be followed.)

```
                    1           1          1
ERROR= YES ,HEADFMT= 2 ,DETAIL1= 2 DETAIL2= 2
                    3           3          3
```

ERROR YES means the next card is the title for the Audit and Error listing.

ERROR NO means no title card is included.

The HEADFMT parameter specifies the number of cards containing the format specification for the heading of the output listing.

The DETAIL1 parameter specifies the number of cards containing the format specification for the first line of the listing for each text word.

The DETAIL2 parameter specifies the number of cards containing the format specification for the succeeding lines of the listing for each text word.

FORMAT CARDS

The formats for the output listing of the text words are specified by variable format cards. The format is keypunched beginning with a left parenthesis, followed by the field specification and closed with a right parenthesis. The field specifications are the standard FORTRAN format specifications and are described in any FORTRAN manual. Columns 1-78 may be used.

The following cards are used:

Title Card--(Optional) For audit listing.

Header Format--(1-3 cards) This is the format for the page heading for the output listing. It should include an integer field specification for the page number and an alphabetic field for the output title card (see below). The heading should not exceed five lines.

DETAIL1 Format--(1-3 cards) This format should provide for the following fields: Frequency of Occurrence (integer), Matchcount (integer), Prefix (8A1), Word (18A1), and the index entries for three text words. Each index entry has four integer fields (for the Prose mode the fields correspond to Volume number, Chapter number, Paragraph number and Word number).

DETAIL2 Format--(1-3 cards) This format provides for writing out index entries for words which occur more than three times in the text. The format should provide for three index entries as in DETAIL1. The Word, Matchcount, Frequency, and Prefix fields will not be written out in this line.

Matchcount Totals Format--(1 card) This format is used to write out the total number of occurrences of each matchcount. Two integer fields should be specified: Frequency and Matchcount.

Output Title Card--(1 card) The contents of this card will be output in the page headings of the output listing. It will be read in under a 13A6 format.

EXAMPLE

```
FUNCT=YES,PUNCT=YES
PRINT=YES0005900059
ERROR=YES,HEADFMT=3,DETAIL1=1,DETAIL2=1
          PUNCTUATION AND FUNCTION WORD LISTINGS
```

```
(H1,15X,13A6,15X,6H PAGE , I6//2X,7HFREQ OF ,2X,5HMATCH,37X,3(25HVOL
CHAP PARA SENT WORD )/3X,5HOCCUR,3X,5HCOUNT,15H  PREFIX    WORD,22X,
3(3HNUM,4(5H NUMB), 2X)/)
(2X,I5,4X,I5,1X,8A1,2X,18A1,7X,3(1H(,I3,2X,I3,1X,I4,2I5,1H)))
((52X,3(1H(,I3,2X,I3,1X,I4,2I5,1H))))
(2H (,I5,4X,I15,1H))
JAMES JOYCE   "THE DEAD"    AUGUST 10, 1971
```

FILES

SUFFIX uses the following mass storage files (magnetic tape, disc, etc.):

Input Text (from INDEX or PREFIX)--FORTRAN logical unit 15.

Output Text--FORTRAN logical unit 13.

Suffix Pairs--FORTRAN logical unit 30.

Function Words and Punctuation Symbols--FORTRAN logical unit 35.

Three Scratch Files--FORTRAN logical units 14, 17, 18.

Parameter Control Cards--FORTRAN logical unit 5.

Output Listings--FORTRAN logical unit 6.

AUDIT AND ERROR LISTING

The Audit and Error Listing has two main parts. The first part is a listing of the parameters that were read in from the control cards. Following the listing of parameters is a listing of the text words. The second part, which follows the text listing, shows the function words and punctuation symbols that were present in the text file. The following error messages may appear:

1. FUNCT REPORT PAGE LENGTH IS GREATER THAN 150. JOB TERMINATED.
   or
   OUTPUT PAGE LENGTH IS GREATER THAN 150. JOB TERMINATED.

   Probable cause: A page length specification exceeded 150 lines.

   The parameter may have been keypunched in the wrong column.

2. CONTROL CARD ERROR IN COLUMN XX. JOB TERMINATED.
   KPR = AAAAAAAAA

   Probable cause: The PRINT parameter specification was misspelled. The nine character alphabetic field contains the word which appeared on the control card. The two digit numeric field contains the number of the column in which the error was detected.

3. CONTROL CARD ERROR IN COLUMN XX. JOB TERMINATED.
   CARD WAS-- (The Control Card Will Be Printed Here)

   Probable cause: A parameter specification other than the PRINT specification was misspelled.

SUFUNA1

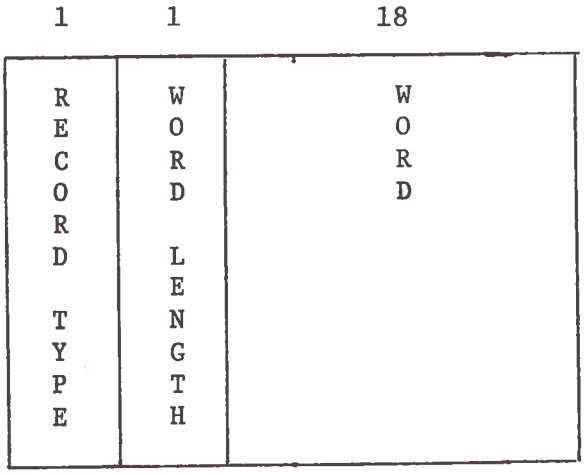
Program SUFUNA1 is used to create the file of function words and punctuation symbols used by program SUFFIX. The input consists of 80 character card images and is read in under the 80A1 format specification. The function word or punctuation symbol MUST begin in column 1 and MUST be followed by at least one blank. The record MUST also contain either the word 'FUNCTION' or the word 'PUNCTUATION', depending on the type of record.

The end of the input file should be marked by placing a card with three exclamation points (!!!) after the last record.

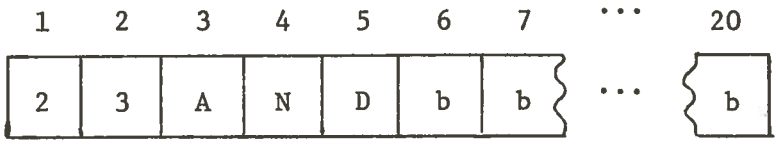
EXAMPLE

```
A   FUNCTION
,  PUNCTUATION
AND FUNCTION
.
.
.
!!!
```

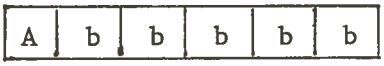
The output records have the following format, with length in computer words (for the Honeywell 635, one computer word contains 36 bits):



If the record is a punctuation record, the type will equal 1. If the record is a function word, the type will equal 2. A type field of 0 indicates end of file. The length field contains the number of characters in the function word or punctuation symbol. The word field contains the punctuation symbol or function word, with one character per computer word starting with the leftmost word in the field. Thus the record for the function word 'AND' would be as follows (b = blank):



NOTE: The Honeywell 635 has 36-bit words. The alphameric character code is BCD with six bits per character. Thus, one computer word can contain six alphameric characters. In the FORTRAN IV VIA package, each computer word usually contains one character in the leftmost position, followed by five blanks (i.e., as in the A1 format). Thus for computer word 3 in the above record the word would contain



.

The Audit report is a listing of the file, showing each word or punctuation symbol and its type code. In addition, the following error messages may appear:

1. INVALID RECORD FORMAT, TYPE FIELD IS INCORRECT. CARD REJECTED.
   Probable cause: The field identifying the record as a punctuation or function record was misspelled.
2. INVALID RECORD FORMAT, TYPE FIELD IS MISSING. CARD REJECTED.
   Probable cause: The field identifying the record as a punctuation or function record was not present.

3. INVALID RECORD FORMAT, BLANK IN COLUMN 1. CARD REJECTED.
   Probable cause: The function word or punctuation symbol did not begin in column 1.
4. INVALID RECORD FORMAT, WORD CONTAINS MORE THAN XX CHARACTERS. CARD REJECTED. Probable cause: The function word had more than the maximum allowable number of characters (currently 18).

Each of the above error messages will be followed by the card image of the offending record. Processing will continue, although the resulting file will probably be incomplete. However, by continuing processing, all the mistakes should be discovered on the first run.

If the input cards were not sorted by alphabetical order (that is, on collating order) the output file should be so sorted before being used by SUFFIX.

SUFN1S

Program SUFN1S is the first of two programs which build the file of suffix pairs and associated exception words and letters. The input file is a deck of cards, or card images, containing the suffix pairs and exception words and letters. The first suffix of each pair must begin in column 1. If column 1 is blank, then the first suffix is the blank. Following the first suffix, there must be at least one blank and then the second suffix. The card containing the suffix pair must be followed by all associated exception records, if any. The word 'EXCEPTION' must begin in column 1, followed by at least 1 blank. If the record is for an exception letter, then the word "LETTER' is key-punched followed by at least one blank and then the exception letter. The exception word should only consist of the letters which precede

the suffix (i.e., "CAPE" and "CAPTION" do not have the same root, thus in the example below the exception word record contains "CAP").

EX.

```
E   TION
EXCEPTION CAP
    Y
EXCEPTION   BUS
```

The last card of the file should contain an explanation point in each of the first three columns (!!!) to indicate end of file.

The output file records are 55 computer words long. The first 18 computer words contain the first suffix of the pair, with one letter of the suffix per computer word. The next 18 computer words contain the second suffix. The next computer word identifies the record type. If the record is the first record for the suffix pair this field contains blanks. If the record is for an exception word, then the computer word contains an 'E' in the leftmost character position (see note below). If the record is for an exception letter the computer word contains an 'L' in the left most character position. The final 18 computer words contain blanks if the record is for a suffix pair. If the record is for an exception word or letter, then the word or letter is contained in this field.

NOTE: Alphameric data are represented in the BCD code which uses six bits per character. Thus, in the 635's 36 bit word, there is room for up to six alphameric characters in each computer word. In this program the characters are stored one per computer word in order to facilitate character manipulation. The character is stored in the leftmost character position followed by five blanks, just as when a letter is read in under the FORTRAN 1A1 format specification.

The output file should be sorted before it is used by program SUFNS2, which finishes building the suffix file. The records may be sorted alphabetically treating the entire 55 computer word record as one field for purposes of the sort.

FILES

Input Records--FORTRAN logical unit 10.

Output File--Written unformatted (i.e., the format of the records are not altered) to FORTRAN logical unit 20.

The input file will probably be cards. The output file should be written to magnetic tape or disc, since it is only an intermediate file.

ERROR MESSAGES

The following error messages may be written out to FORTRAN logical unit 6 (the printer):

1. EXCEPTION WORD MISSING. CARD REJECTED.
   Probable cause: The card contained the word 'EXCEPTION' but then did not contain an exception word.
2. EXCEPTION LETTER MISSING. CARD REJECTED.
   Probable cause: Similar to the above case.
3. EXCEPTION LETTER HAS TOO MANY CHARACTERS. CARD REJECTED.
   Probable cause: The record is an exception letter record, but there was more than one character where the letter should have been.
4. TOO MANY CHARACTERS IN FIRST SUFFIX FIELD. CARD REJECTED.
   Probable cause: The first suffix of the pair had more than the maximum allowable number of characters (currently 18).
5. TOO MANY CHARACTERS IN SECOND SUFFIX FIELD. CARD REJECTED.
   Probable cause: Similar to the above case.

6. SECOND SUFFIX MISSING. CARD REJECTED.

   Probable cause: The first suffix of the pair was present, but there was no second suffix. If the second suffix was intended to be the blank, it must be the first member of the pair.

7. NO SUFFIX FOR EXCEPTION RECORD. CARD REJECTED.

   Probable cause: If the card containing the suffix pair is rejected for one of the above reasons, then the associated exception records will also be rejected.

All of the above messages will be followed by a listing of the card that caused the message to appear.

SUFNS2

Program SUFNS2 accepts the sorted file of suffix pairs and associated words and letters which was created by SUFN1S. It builds tables of suffixes, exception words and letters, and associated pointers in the format that will be used in the SUFFIX program.

The input file is read in binary (i.e., no change is made to the format of the input records) from logical unit 20. This file should have been written to magnetic tape or disc by program SUFN1S and sorted by alphabetical order.

The output file is written in binary to logical unit 30. This file should be written to magnetic tape or disc for use by program SUFFIX.

The following error messages may be written to logical unit 6 (the printer):

1. ILLEGAL TYPE FIELD, X, FOR UPDATE RECORD. JOB TERMINATED.

   Probable cause: The field indicating the type of a record, suffix pair or exception record, was incorrect. This can only happen if there is a programming error in this program or in SUFN1S.

2. ILLEGAL TYPE FIELD IN RECORD FOR NEW SUFFIX. JOB TERMINATED.

   Probable cause: An exception word or letter record was encountered for a new suffix pair before the record for the suffix pair itself was encountered. Check the listing from program SUFN1S to make sure the file was created properly (i.e., no input cards were rejected by SUFN1S). Next, make sure the file was sorted correctly. Finally, check for a program error.

3. LIMIT OF XXXXX SUFFIX 1 RECORDS EXCEEDED. JOB TERMINATED.

   Probable cause: Currently, there may be no more than 150 different first suffixes. If it is necessary to allow more than 150 first suffixes the dimensions of the following tables must be increased in Programs SUFFIX and SUFNS2: KSUFF1, NOSUF1, INDSF2, ISFCN2. In addition, the value for MAXSF1 must be increased in the DATA statement to which it is initialized.

4. LIMIT OF XXXXX SUFFIX 2 RECORDS EXCEEDED. JOB TERMINATED.

   Probable cause: Similar to the previous message. The current maximum for second suffixes is 375. To increase this maximum, the dimensions of the following tables must be increased in both programs SUFFIX and SUFNS2: KSUFF2, NOSUF2, IEXPIN, IEXPCT, ILETIN, ILETCT. In addition, MAXSF2 must be initialized to the new value in the DATA statement in which it is initialized.

5. LIMIT OF XXXXX EXCEPTION WORD RECORDS EXCEEDED. JOB TERMINATED.

   Probable cause: Similar to message number 3. The current maximum for exception records is 800. To increase this maximum, tables IWORDX and IWRDXL must be given increased dimensions in both programs SUFFIX and SUFNS2. In addition, the DATA statement which initializes MAXEXP must be altered to reflect the new maximum.

6. LIMIT OF XXX EXCEPTION LETTER RECORDS EXCEEDED. JOB TERMINATED.

   Probable cause: Similar to message number 3. The current maximum for exception letter records is 50. To increase this maximum, the dimension of table LETXCP must be increased in both programs SUFFIX and SUFNS2. In addition, the DATA statement which initializes MAXLET must be altered to reflect the new maximum value.

IMPORTANT: Program SUFNS2 uses subroutine COMPCT which is a character manipulation routine to be supplied by the user. Consult the section of this manual dealing with implementation considerations.

A. 6. SUFFIX PROGRAMMER'S MANUAL

by Frank Joyce

SUFFIX is composed of six modules, or subroutines: SUFFX1, SFINIT, SFREAD, SFOUT1, SFMATC and SFINAL. Each of these modules is described in detail below. SUFFIX is written in FORTRAN IV and requires approximately 25,000 words of storage space on the Honeywell 635 computer.

SUFFX1

SUFFX1 is the main logic module for the SUFFIX program. This module builds a table of content words which will be sent to module SFMATC for identification of common roots. The table contains all the content words which begin with the same three letters. When a content word is encountered which does not begin with the same three letters as the words in the table, the table is sent to SFMATC. After SFMATC identifies the words which have common roots, SUFFIX1 calls routine SFOUT1 to write out the words and index information (see SFOUT1 below). Next, the table is reinitialized for a new group of words and the above process is repeated. When the text file has been completely processed, module SFINAL is called in to post process function and punctuation records and print out a final summary of the run.

The first step of SUFFX1 is to read in initial tables and control information. Module SFINIT is called to read in the user supplied parameter cards (see SFINIT below and the user's manual section of this report). Next, module SFREAD is entered through a call to entry point SFRDIN. The purpose of this call is to read in the first text word. Finally, the suffix pairs and exception words are read in. This is accomplished by entering module SFMATC via entry point SFMTC1.

The initialization process is completed by placing the first three letters of the first content word in array IRTSAV and setting counter ISAVEC equal to one. ISAVEC is used to record the number of words placed in the table.

SUFFX1 now begins the main processing function with a call to module SFREAD. The text file has been sorted into alphabetical order prior to the execution of SUFFIX. The call to SFRDIN caused the first record for the first content word to be read in. The word and associated index information was placed on a scratch file and a copy of the word was passed to SUFFX1. SFREAD now reads in the remaining records for this current content word and places the index information on the scratch file. SFREAD then reads in the first record for the next content word (any intervening function words or punctuation marks are written out to scratch files) and returns control to SUFFX1. At this point, array INWORD contains the text word which is being processed, and array NEWORD contains the next word to be processed.

Next, plural and possessive forms are removed from the word in array INWORD. This section of the module removes "'s", "'", "s", or "s'" (if the word does not end in "ss"), and "es" or "es'" (if the "es" is preceded by "s" as in "guesses" or "buses") from the end of the text word for purposes of root grouping. The actual text records, which are on a scratch file, are not altered by this procedure.

The word is now ready to be entered in the root grouping table, IRWORD. First, a check is made to assure that the word has more than three letters (no English words have been encountered which have legitimate suffixes and are not longer than three letters). Next, the word is transferred to the root grouping table. This is accomplished

using routine COMPCT. The array INWORD stores the text word one letter per computer word. In order to save storage space, the text word is packed into the table at maximum character density (see the section of the user's manual entitled "IMPLEMENTATION CONSIDERATIONS" for a more detailed discussion of the role of COMPCT). The text word in the table begins with the fourth letter of the actual text word, since the first three letters can be found in array IRTSAV. As SFREAD read in the text records, a count was maintained in counter NCOUNT. Its value is now moved to array ICOUNT which is associated with the root grouping table.

The new word in array NEWORD is compared against the three letters in array IRTSAV to see if the new word also belongs in the table. If it does, the above procedure is executed again beginning with the call to SFREAD. If the first three letters do not match the words already in the table, the module SFMATC is called to assign matchcounts to the words in the table. SFOUT1 is then called to write out the records to the output file and the printed report. Finally, IRTSAV is reinitialized to 1. The main processing loop is then reentered.

SFINIT

Module SFINIT is used to read in the control parameter cards described in the user's manual.

SFREAD

Module SFREAD reads in text records, identifies function word and punctuation records, and passes content words to module SUFFX1.

The first step is to move the content word read in previously from array NEWORD to array INWORD. The word is written out to FORTRAN logical unit 14 along with its index information. A loop is then entered

which reads in the remaining records for this word. Only the index information for these records is written out to the scratch file. NCOUNT is used to keep track of the number of records for the word.

When a record is encountered for a new word, then the word is checked against the file of function words and punctuation records. If the word is a function word or punctuation symbol, it is written out to FORTRAN logical unit 18. The remaining occurrences of the word are read in and the index information is written out to file 18. KCURR is used as a counter for the occurrences. When all occurrences of the word have been written to file 18, KCURR and the type of the record (function or punctuation) are written out to FORTRAN logical unit 17.

The next text word is then checked against the function word file as above. When a content word is encountered it is placed in array NEWORD and control returns to module SUFFX1.

SFMATC

Module SFMATC processes the contents of array IRWORD, assigning a common matchcount to words which have a common root. Array IRWORD is the table of content words for which the first three letters are identical that was build by module SUFFX1. Array IRMATC will be used to record the matchcounts for the table entries. A check is made for a single content word entry in IRWORD and if present, it is assigned a unique matchcount value. Counter MATNUM, which keeps track of the matchcount, is incremented by 1 and control returns to module SUFFX1.

If more than one entry is present in IRWORD, the content word at the beginning of the table (indexed by I) is compared with the word at the end of the table (indexed by J). The search is made for content words which do not yet have a matchcount, i.e., which have not been

paired with another word(s). If the content word (I) has no matchcount, it is assigned the value in MATNUM. If it does have a matchcount (i.e., IRMATC(I) does not equal zero) then I is incremented by one and the test is repeated. Once a word (I) has been found without a matchcount, a search is made for a content word (J) which has not been assigned a matchcount, decreasing J until such a word is found. An attempt is now made to match word (J) with word (I). If word (J) matches word (I), then the matchcount of the word indexed by J is set equal to the matchcount of word (I). These words are considered to have a common root and matchcounts of common roots are equal. The process of decrementing J and comparing continues until J is equal to I. At this point, if I does not equal the total number of entries in IRWORD, I is incremented by one and J is reset to the last table entry. Once again, I proceeds forward through the table until reaching another content word which has no matchcount. MATNUM is incremented and the new value is assigned to the new word (I), and the above matching process is repeated. All content words have been processed when I exceeds the number of entries in IRWORD. Control then returns to module SUFFX1.

The process of comparing succeeding words in IRWORD for possible suffixes and common roots is accomplished by employing tables of legitimate suffix pairs and associated exception words. Words (I) and (J) are stored in table IRWORD at maximum character density. The Honeywell 635 has 36-bit computer words and employs the BCD code for representation of alphameric characters. Since the BCD code employs six bits per character, a maximum of six characters may be stored in each computer word. Since the current maximum length for text words

is 18 letters, each text word may be stored in three computer words. This allows a substantial saving of space in storing table IRWORD, which can hold up to 500 content words. Thus, 1500 computer words are required for table IRWORD (500 x 3) as opposed to the 9000 computer words required if the text words were stored with one letter per computer word (500 x 18). A similar saving is achieved in the tables of suffixes and exception words. However, a price must be paid in operating efficiency. In order to perform the necessary character manipulations in FORTRAN, it is necessary to "unpack" the text words to be examined. This is accomplished by using subroutine BRKUP (see the section of the user's manual entitled "IMPLEMENTATION CONSIDERATIONS" for a description of BRKUP).

Words (I) and (J) are moved to arrays KWORDI and KWORDJ respectively using subroutine BRKUP. The two content words are checked for equality, letter by letter. If equal, the words should obviously be grouped as a match. If the words are not equal, the letter by letter comparison will find the point of deviation. The remaining portion of KWORDI is transferred to array MSUFF1 and the remaining portion of KWORDJ is transferred to MSUFF2. MSUFF1 and MSUFF2 contents are compared and switched, if necessary, so that MSUFF1 will contain the lowest value in collating sequence. This is necessary because the tables containing suffixes have been sorted so that the smaller of the suffix pair occurs in the ISUFF1 (first suffix) table.

A binary search is made in the ISUFF1 table using the contents of MSUFF1 (potential suffix from text word) for the argument. If the suffix is found, an entry is made to the ISUFF2 table based on the starting address found in the array INDSF2 associated with the first

suffix. A binary search is made within the associated second suffixes using the contents of MSUFF2 as the argument. Array ISFCN2 contains the number of second suffixes associated with the first suffix. If a match is found in ISUFF2, a test is made for the LETTER exception. If this rule is in effect, the last letter which the two text words had in common must match one of the letters in array LETXCP associated with the suffix pair. The value in array ILETCT associated with the ISUFF2 entry will be equal to zero if the LETTER rule does not apply. Otherwise, it will contain the number of associated letters. A comparison is made using the last common letter before the point of deviation was reached. Entry is made to the LETXCP table using the address found in array ILETIN. If the letters are the same, the LETTER rule is satisfied, the suffix pair is still legal, and a further search of the list of exception words is warranted. If the letters are not equal, the next letter is checked. This process continues until either a match is made or the number of letters indicated in ILETCT have been checked. If the LETTER rule applies but there is no match between letters, then the words are not matched.

Assuming the LETTER rule was either satisfied or not in effect, a binary search is made of the exception words associated with the suffix pair. The exception words are contained in array IWORDX. The beginning address of the exception words associated with the suffix pair is contained in array IEXPIN and the number of associated exception words is contained in array IEXPCT. Before testing for an exception word, it is necessary to construct the text word since table IRWORD does not contain the first three letters of the word. The word is reconstructed in array IWRDCK. First, the three letters in IRTSAV are

moved to IWRDCK, then the letters of the text word are moved until the point of deviation is reached. The exception word should consist only of the letters which precede the suffix, i.e., "CAPE" and "CAPTION" do not have the same root, thus the exception word would be spelled "CAP". If the text word is not found in the exception list, then the suffix pair is legitimate and the words are matched. If the text word does match an exception word, then the words do not match.

ALTERNATE SFMATC

An alternate version of SFMATC has been developed which employs assembly language routines to do character manipulations of strings of characters packed six per computer word.

Subroutine KOR09A is a character string routine which moves an arbitrary number of characters from an arbitrary source string to an arbitrary destination string. Calling sequence:

```
CALL KOR09A(NMOVE,SOURCE,NSOUR,DESTIN,NDEST,IERROR)
```

where

NMOVE is the number of characters to be moved
SOURCE is the name of the source string
NSOUR is the position of the first character to be moved relative to the beginning of SOURCE
DESTIN is the name of the destination string

NDEST is the destination position of the first character relative to the beginning of DESTIN
IERROR is an error flag (not checked by this version of SFMATC).

Subroutine KOR21A is used to compare two strings of a variable number of characters, character by character. Calling sequence:

```
CALL KOR21A(N,STRNG1,NCHAR1,STRNG2,NCHAR2,RESULT,IERROR)
```

where

N is the number of characters to be compared
STRNG1 is the first word address of the first string
NCHAR1 is the starting character position relative to STRNG1.

STRNG2 is the first word address of the second string
NCHAR2 is the starting character position relative to STRNG2
RESULT is the result of the comparison
IERROR is an error flag (not checked by this version of SFMATC).

Comparison results:

RESULT = -I; The Ith character in STRNG1 is less than the Ith character in STRNG2
RESULT = 0; All characters are equal
RESULT = I; The Ith character in STRNG1 is greater than the Ith character in STRNG2.

SFOUT1

Module SFOUT1 produces a listing of content words, associated frequency and matchcount data, and a listing of index information. It also writes the content records out to the output file.

SFOUT1 begins by accumulating the frequencies for each matchcount. In the process, several tables are employed. Table IRMATC contains the matchcount assigned to each word type by routine SFMATC; table ICOUNT contains the number of times each word occurred in the text; table MCFREQ is used to accumulate the total frequency for each matchcount; and table MPOINT links each word to its matchcount total in MCFREQ. Data item MCTEST is used to keep track of the current matchcount and integer M is used as an index. MCTEST and M are initialized to zero before entering the accumulating loop. The accumulating loop uses I as an index for running through the words grouped by SFMATC. The matchcount (I) is compared to MCTEST. If it is less than or equal to MCTEST it has already been included. If it is greater than MCTEST, then M is incremented by one and MPOINT(I) is set equal to M, MCTEST is set equal to the matchcount (I), and MCFREQ(M) is set equal to

ICOUNT(I). The rest of table IRMATC is searched for matchcounts equal to MCTEST. When one is encountered, its count is added to MCFREQ(M) and the corresponding MPOINT(I) is set equal to M. When all the equal matchcounts have been found, M is incremented by one and the loop is executed again.

Next, summary records are written for the matchcount frequencies, assuming the written report was requested by the user. These records consist of the frequency and matchcount for one matchcount at a time and are written according to a user supplied format (see user's manual).

Finally, the scratch file (FORTRAN logical unit 14) containing the content words is rewound and the first word read in. Table ICOUNT contains the frequency count for each content word, and thus may be used to control the number of index records read back in. The output file records are written to FORTRAN logical unit 13. The printed report records are written to logical unit 6 using user supplied output formats. The first line for each content word contains the word's frequency, matchcount, prefix (if any), the word itself, and index information for up to three occurrences of the word in the text. Succeeding lines for that content word will contain only the index information. After the scratch file has been emptied, the file is rewound to allow the next group of content words to be written to it and control returns to module SUFFX1.

SFINAL

Module SFINAL is called after the text file has been processed. It writes the function word and punctuation records to the output file, lists the function words and punctuation symbols and prints a summary. The function words and punctuation symbols have been stored

on scratch file 18 along with associated index information. Scratch file 17 contains a frequency count for each record and a field indicating whether the word is a punctuation or function record. The module reads the two fields from file 17 and the function or punctuation record from file 18. The frequency field, KCURR, determines how many index records will be read in from file 18. The output records are written to file 13 and the function word or punctuation symbol is written to the printed report along with KCURR. The index information does not appear on the printed report. The next record is then read from both file 17 and file 18 and the above process repeated.

When all the punctuation and function word records have been processed, a brief summary of the run is printed out using counters which have been maintained in the various modules. These counters are INCNT, which records the total number of text records read in; JOTCNT, which records the number of punctuation and function records written out; ICNTNT, which records the number of content word records written out; and KTOTAL which is the sum of JOTCNT and ICNTNT. Of course, KTOTAL should be equal to INCNT.

A. 7. LIST-Structure THESAUR

by Peggy Lewis

INTRODUCTION

The VIA package for text processing includes capabilities for text selection and list-structured thesaurus output. By means of the program SELECT, processed text (output from SUFFIX) may be selected according to location by volume, chapter, paragraph, and sentence. The output file which is produced may then be introduced as data to the list-structured thesaurus outputting program (THESR). Alternately, the entire selection procedure may be bypassed, and the SUFFIX output introduced directly to THESR as input.

The THESR program subroutines will prune a thesaural tree which is defined to a maximum of five levels and described as a list of ordered pairs so that the thesaural words which appear in the output tree, or words with the same match count, occur in the text. Several run-time processing options, which are descrived later, are available. At this time, the input thesaurus must be provided by the user.

SELECT

I. INTRODUCTION

Whenever it is desirable to process only a portion of the records from a given SUFFIX output file, the SELECT program may be used to create an entirely new data file containing only those records. SELECT is run as an independent activity with information relative to the selection procedure supplied by the user on file SYSIN (System standard input, usually cards).

In order to preserve the indexing information for each record, the records which are selected are copied directly to the output file; they will not be changed in any way. Thus, the frequency count fields will NOT reflect frequency of occurrence within the newly-created file. The VIA program (THESR) which produces list-structured thesaurus output will calculate frequency counts itself, hence output from SELECT may be introduced directly as input to that program. If for any reason it is necessary to supply the SELECT file records with accurate frequency counts, this must be done as a separate activity. (SEE: NOTES ON FREQUENCY COUNTS)

The input and output file descriptions below will complete the documentation necessary for normal use of SELECT. Some additional notes for programmers will follow the file descriptions.

II. FILE DESCRIPTIONS

Each file will be referenced by its variable name within the program as well as the logical unit number which is assigned to it. The logical unit numbers should link the files to their physical devices by means of the job control language for the computer being used.

ISUFX (Input File)

Logical Unit Number: 1

Device Type: Tape, Disc, or Drum

This file is output from the SUFFIX program or a previous SELECT run. It is written in binary mode. Each logical record contains fifteen fields, a total of 39 computer words, as follows:

	Word No.	Contents
	1	Line
Index for Comparisons	2	Volume
	3	Chapter
	4	Paragraph
	5	Sentence
	6	Word-in-sentence
	7	Page
	8	Idiom Flag
	9	Match Count
	10	Match Count Frequency
	11	Type Frequency
	12	Prefix Length
	13	Word Length
	14-21	Prefix
	22-39	Word

All except the Prefix and Word fields are integer values. Prefix is a field of eight computer words, with one character stored left-justified within each word. If the prefix is less than eight characters in length, it is left-justified within the field. Word is a field of eighteen computer words, stored one character per word, left-justified, as with the prefix.

Records need not be introduced in any particular order for this program to process them successfully.

ITEXT (Output File)

Logical Unit Number: 2

Device Type: Tape, Disc, or Drum

The TEXT output file is also written in binary mode. The records which are written to this file are only those which match the indexing parameters which are specified by the user. Each record has exactly the same form as the records which are read from ISUFX. (SEE: ISUFX)

SYSIN (Input File)

Logical Unit Number: 5

Device Type: System Standard Input (Assume card reader)

The SYSIN data cards should begin with a 'SELECT' card which designates the level of index processing desired:

```
SELECT =   V     (The level corresponding to Volume)
           C     (The level corresponding to Chapter)
           P     (The level corresponding to Paragraph)
           S     (The level corresponding to Sentence)
```

Cards immediately following the 'SELECT' card should contain indexing information in columns 1-72. Either single or ellipsis notation may be used.

Single Notation:

Each group of level parameters is enclosed in parentheses, and the parameters within the group are separated by commas. The number of parameters should always coincide exactly with the processing level on the 'SELECT' card.

Ellipsis Notation:

Two single groups of parameters are enclosed in a third set of parentheses and separated by a comma. Text bracketed inclusively by the parameter groups will be processed; it is therefore important that the "smaller" of the index specifications should come first in the pair.

EXAMPLE:

To specify selection of Volume 1, Chapter 5, Paragraph 23, and Volume 1, Chapter 6, Paragraphs 7 through 15:

```
SELECT = P
(1,5,23), ((1,6,7),(1,6,14))
```

Card columns 1 through 72 will be processed for each card; blanks will be ignored. An asterisk in column 72 of a specification card indicates data continuation on the next card. All parameters must be separated by commas.

III. NOTES FOR PROGRAMMERS

MAJOR ROUTINES

Flowcharts for the major processing routines (DECODE and PICKUP) are included. Briefly, DECODE reads the SYSIN file and builds an array of indexing information for later comparison to the appropriate fields of each ISUFX record. This array (ARRAY) is doubly-subscripted. The first subscript is set to five; four places are reserved for the four possible levels associated with each index, and a fifth position is reserved for a flag which indicates whether it is single notation (a value of 0) or ellipsis notation (a value of 1) for that index. The variable LEVEL will be set to the number of levels indicated by the user for processing. The second subscript is arbitrarily set to 25, for a maximum of 25 selection indices per run. This may be changed to any other convenient value, as long as the value is also assigned to the variable MAX. The variable NEXT will be set to the number of indices actually recorded for each run.

Labeled common storage area PARM is used for communication between DECODE and PICKUP; any changes made should apply to both subroutines.

PICKUP reads a record from ISUFX, checks index information in fields 2-5 against the array to the appropriate level, and either outputs the record to ITEXT or ignores it and proceeds to read the next ISUFX record. Processing ends when the end of file is encountered on ISUFX.

SPECIAL-PURPOSE ROUTINES

LCOMP(IWORD1,IWORD2)

LCOMP is a function subroutine which logically compares two words. All comparisons are in the standard collating sequence. The value returned for LCOMP is as follows:

-1	Word 1 logically less than Word 2
0	Word 1 equal to Word 2
1	Word 1 logically greater than Word 2

(LCOMP is a library routine for the Kansas University HW 635.)

IBNRY(IWORD)

IBNRY is a function subroutine. IWORD is a one-digit number in character representation (A1) left-justified in the word. IBNRY converts this number to the normal integer representation, which becomes the value returned by the routine.

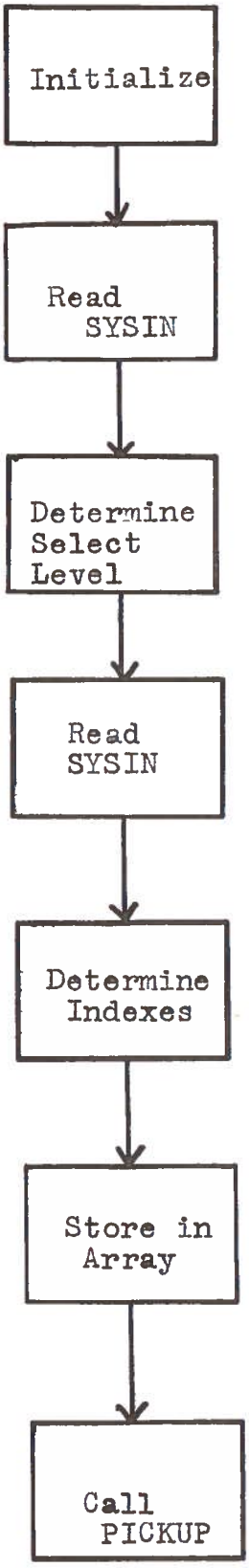
NOTES ON FREQUENCY COUNTS

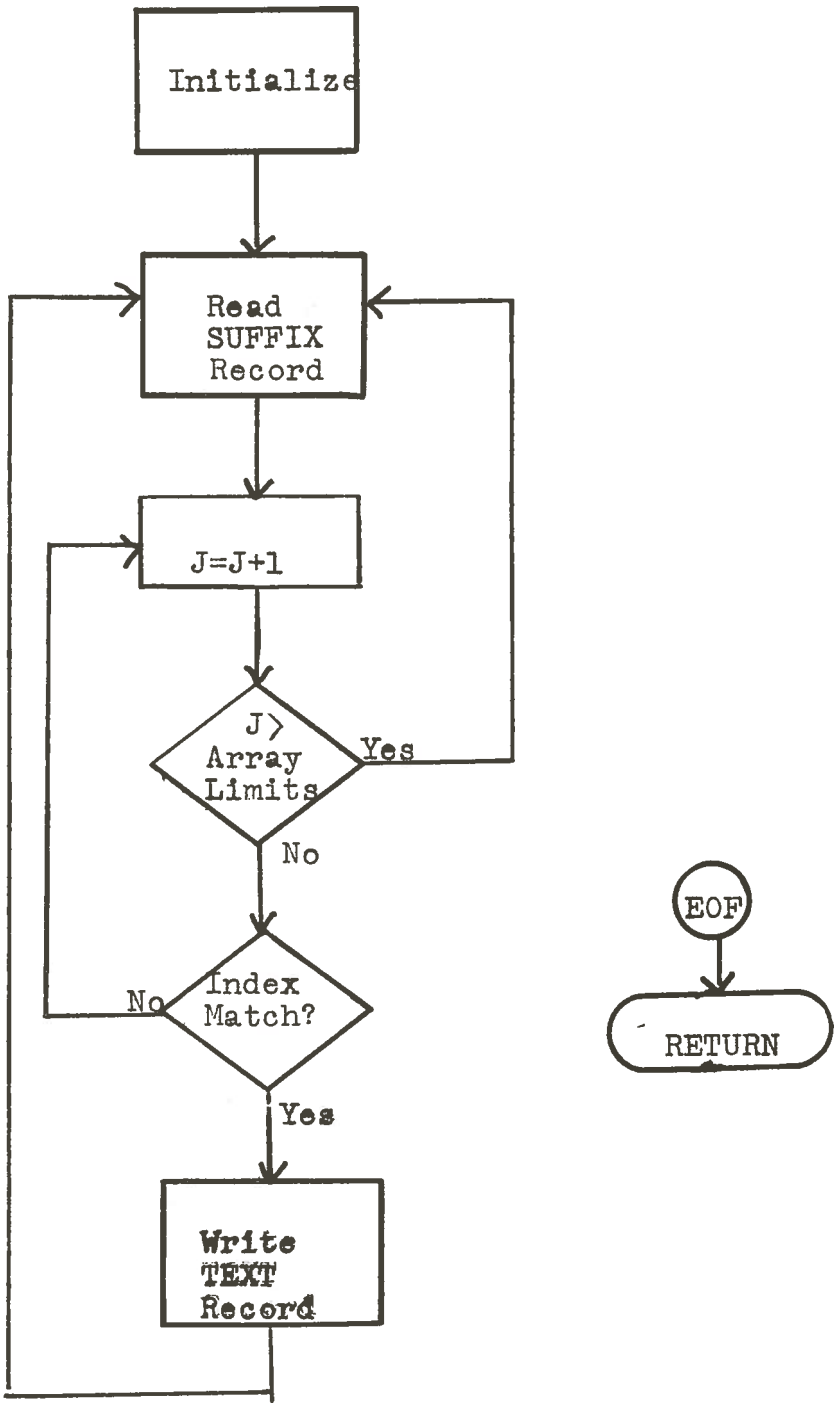
The flowchart for a program which would determine new Match Count and Type frequencies for ITEXT is included*. It could be written and run as a separate activity or assimilated into the PICKUP subroutine without a great deal of difficulty. It would require that the input file be sorted on Match Count, and within Match Count, sorted alphabetically. Because THESR (the assumed destination for output from

---

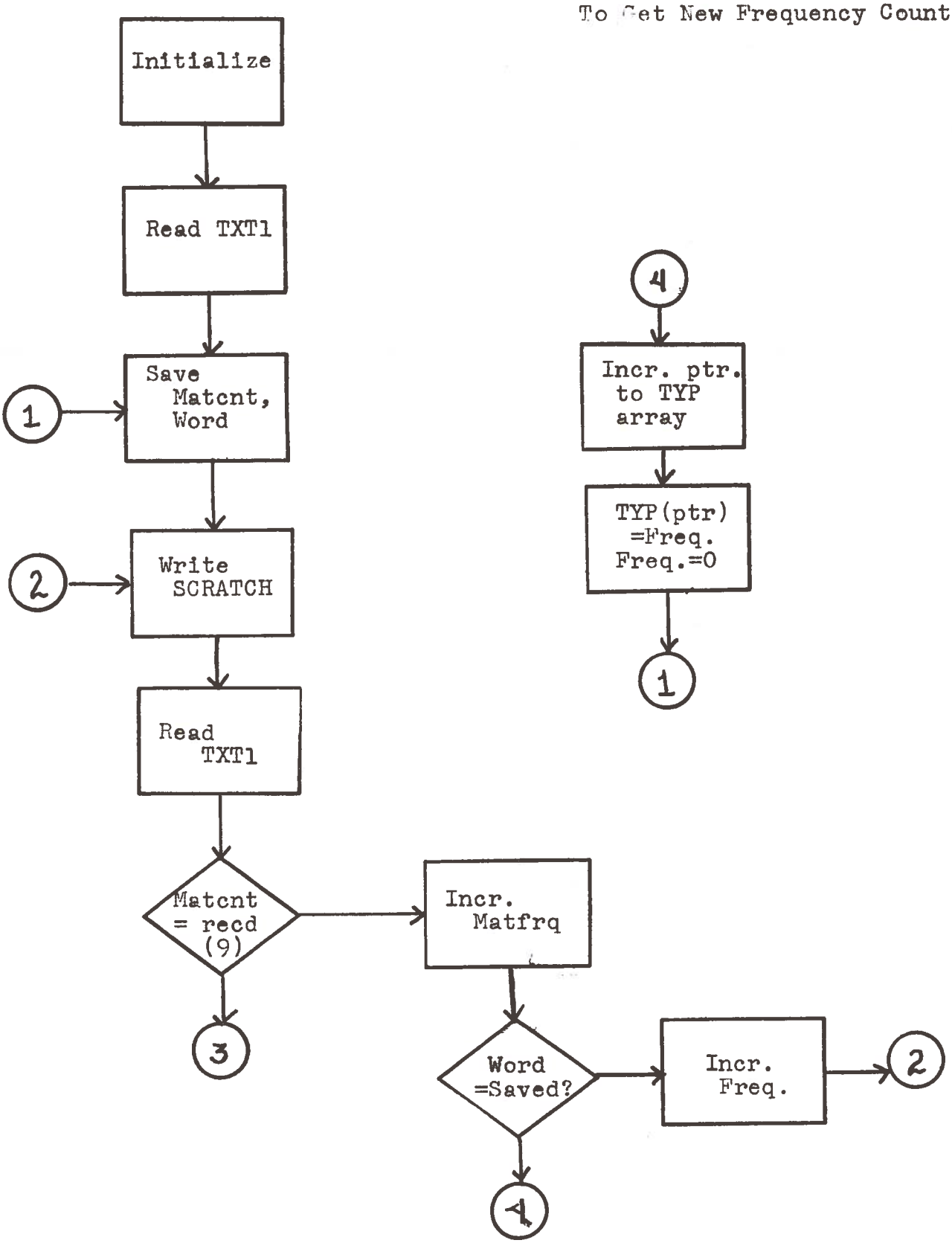
* pp. 156-157.

SELECT) performs its own frequency counting, implementation of this routine would be useful only to those who would want the indexed text sections for purposes other than VIA processing.

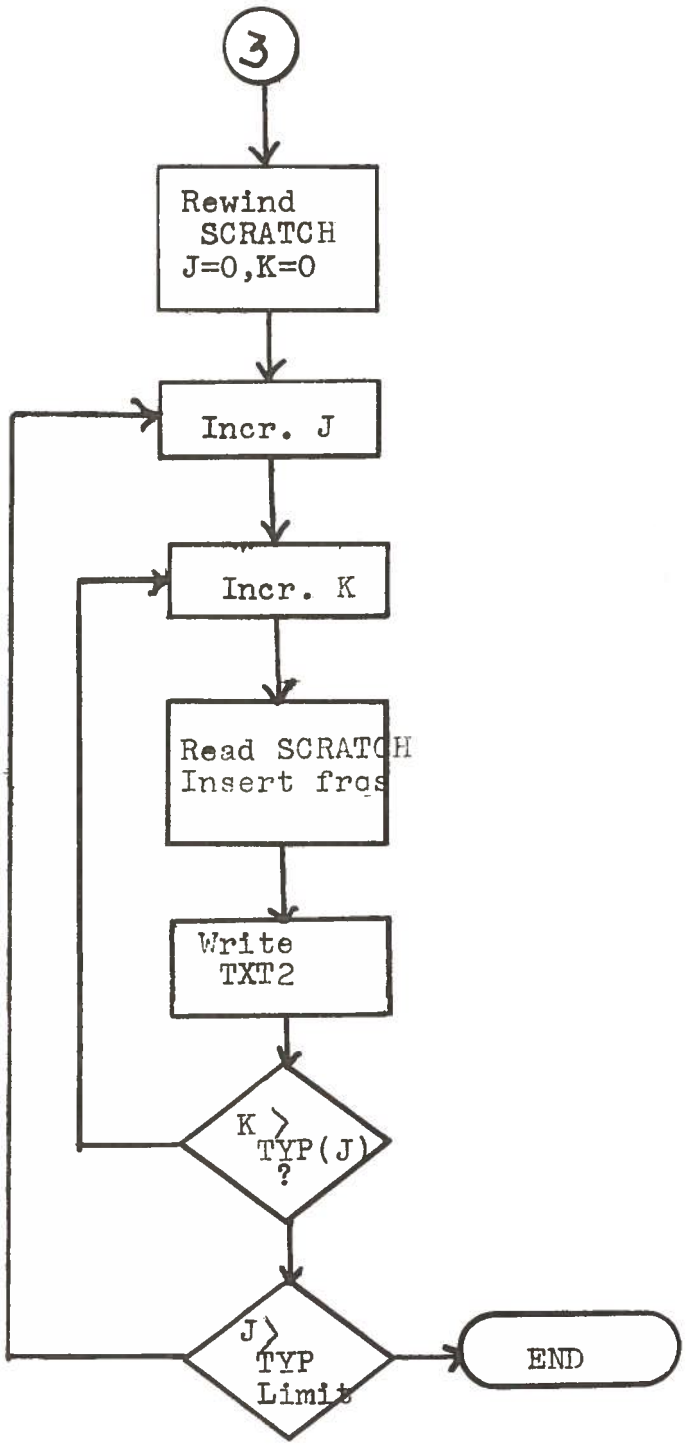

Initialize
Read SYSIN
Determine Select Level
Read SYSIN
Determine Indexes
Store in Array
Call PICKUP


Initialize
Read SUFFIX Record
J=J+1
J> Array Limits
Yes
No
Index Match?
No
Yes
Write TEXT Record
EOF
RETURN

To Get New Frequency Counts



To Get New Frequency Counts
(Page 2)



THESR

I. INTRODUCTION

The THESR program is part of the VIA package for text processing. It consists of a series of processing subroutines which read primary and associate word pairs from a thesaurus file (THES) and compare them to a file of text words (TEXT). When both words of the pair are found in the text file, they are put into a temporary thesaurus array. Associate words which also appear as primaries are linked to the appropriate primary with a pointer. From this information, THESR subroutine THESR4 prepares a linked list of thesaurus words for output to SYSOT (The system standard output file).

Several processing options are available. The level to which the data structure is output may be specified up to and including a level of five. This linked list may also be saved for later comparisons to other runs. The primary and associate word pairs which are specific to the processed text may be saved separately (perhaps for later use as the input thesaurus to process a separate section of the same text). The page heading and formats for outputting the linked list may be specified by the user. (SEE: Input file SYSIN for complete specifications.) Finally, a utility sort may be substituted for the sorting subroutine which is provided. (SEE: SORT SUBSTITUTION for complete specifications.)

The input and output file descriptions and the procedure for substituting a utility sort will complete the documentaion necessary for normal use of THESR. Some additional notes for programmers follow the users' documentation.

II. FILE DESCRIPTIONS

Each file will be referenced by its variable name within the THESR common region labeled FILES as well as the logical unit number which is assigned to it. The logical unit numbers should link the files to their physical devices by means of the job control language for the computer being used.

THES (Input File)

Logical Unit Number: 1

Device Type: Tape, Disc, or Drum

Each logical record in this file is a word pair, consisting of an 18-character primary word field followed by an 18-character associate word field. The characters are stored one per computer word, left-justified with blank fill to the right. The primary or associate word which does not fill the field is left-justified with blank fill to the right. This file is written in binary mode.

Word No.	Contents
1-18	Primary word
19-36	Associate word

TEXT (Input File)

Logical Unit Number: 2

Device Type: Tape, Disc, or Drum

Output from the SUFFIX program or the SELECT routine is used as input to THESR. (SEE: SUFFIX or SELECT program documentation for a complete description.)

SYSIN (Input File)

Logical Unit Number: 5

Device Type: System Standard Input (Assume card reader)

The SYSIN file should begin with a parameter card as follows:

LEVEL = [2 / 3 / 4 / 5] , MICRO = [YES / NO] , LIST = [YES / NO] , FORMATS = [YES / NO]

Brackets indicate a choice. Blanks will be ignored, however all four parameters must appear on the card, always in this order, and always with the equal signs.

LEVEL specifies the number of levels of linkage desired for the list-structured output. The number of levels cannot exceed five.

MICRO specifies whether a text-specific micro-thesaurus will be produced as output on logical unit 4. (SEE: MICRO output file documentation.)

LIST specifies whether the list-structured output will be saved as a permanent file. If LIST = YES, the data will be output to logical unit 3, with a level index added for convenience. (SEE: LIST output file documentation.)

FORMATS specifies whether a header label and variable format data cards are to be read from SYSIN. If FORMATS = NO, a default heading and a set of default formats will be used; no further data cards will be read from SYSIN.

If FORMAT = YES, the card which immediately follows the parameter card will contain header information in columns 1-79, exactly as it is to appear in the output. Column 80 is reserved for an asterisk to indicate continuation of the header on the next card; any other character in that position will be ignored. In no case will more than 132 columns of information be retained for the header.

Following the header information will be pairs of data cards which contain formats for each level of THESR output. Each card contains standard FORTRAN IV format specifications which are appropriate to the level and type of output, and which are enclosed by parentheses. (Note: The data card will look exactly like a standard FORTRAN IV format statement, without the statement number and the word 'FORMAT'.)

Exactly one format will be read from each card, and beginning with level one, the card pairs will be assigned to sequential levels, up to the number of levels which were specified on the parameter card. None of the formats may be omitted for levels less than or equal to the specified level.

The first card of each pair must provide the format specifications for an integer-valued 'Match Count' field, followed by an integer-valued 'Frequency' field, followed by NWDS computer words of packed characters. The second card, which specifies the format for alternate words on the same level, must provide only for NWDS computer words of packed characters. (NWDS is a variable in the program which is set equal to the number of words which will be required to hold 18 characters of information, packed.)

EXAMPLE:

Suppose the level of linkage specified is 3. A header label with 3 level headings is to appear with the output; six format cards will be necessary. Following the parameter card in the SYSIN file these cards will produce the desired result:

(Col. 1)

^^MATCH^C1^^FREQ^^LEVEL^1^^LEVEL^2^^LEVEL^3

(I10, I6, 2X, 3A6)
(18X, 1H*, 3A6) } Format specifications for level 1

(I10, I6, 11X, 3A6)
(27X, 1H*, 3A6) } Format specifications for level 2

(I10, I6, 20X, 3A6)
(36X, 1H*, 3A6) } Format specifications for level 3

SYSOT (Output File)

Logical Unit Number: 6

Device Type: System Standard Output (Assume Printer)

The 5-level linked list structure will appear on the SYSOT file (system standard output file). Each word will appear with its match count, frequency, and alternates. Levels of linking will be indicated by different formats for each level. The formats may be specified at run time. (SEE: SYSIN input file documentation.) The formating which is supplied by the program, if not overriden, will indicate subsequent levels by deeper indentation from the left hand side of the output page.

EXAMPLE: (A sample of possible output from THESR)

```
MATCNT   FREQ    LEVEL 1    LEVEL 2    LEVEL 3

   23     7       ACT
                 (ACTING)
                 (ENACTED)

   49    12                   DRAMA
                             (DRAMATIZE)
                             (DRAMATIC)
```

LIST (Output File)

Logical Unit Number: 3

Device Type: Tape, Disc, or Drum

This optional output file will contain the linked list of text-specific thesaurus words which are written to SYSOT, in the same order that they are written to SYSOT. In addition to the word, the match count, and the frequency count for each word, a level index number is added to the record. Alternate forms of thesaurus words which appear in the text will appear in LIST with their level and match count fields specified, but the frequency field for each will be set to zero. This file is written in binary mode. Logical records are specified as follows:

Word No.	Contents
1	Level index number
2	Match count
3	Frequency
4-6	Word

(Note: The last field is NWDS in length. This length may differ between computer installations.)

EXAMPLE:

For the section of SYSOT output included in the last example, the corresponding LIST output would be:

(Word 1)	(Word 2)	(Word 3)	(Words 4-6)
1	23	7	ACT
1	23	0	ACTING
1	23	0	ENACTED
2	49	12	DRAMA
2	49	0	DRAMATIZE
2	49	0	DRAMATIC

MICRO (Output File)

Logical Unit Number: 4

Device Type: Tape, Disc, or Drum

This optional file contains a text-specific thesaurus of word pairs, written in a manner exactly analogous to the THES input file. Each record will contain a word pair; each word of the pair will occupy a

field of 18 characters which are stored one per computer word, left-justified within the computer word, and left-justified within the field. The file is written in binary mode.

III. SORT SUBSTITUTION PROCEDURE

In order to substitute a system utility sort activity for the somewhat clumsy sorting subroutine which is provided, this procedure should be followed:

The mainline THESR and the SORTIT subroutine should be ignored, and the programs SUB1, SUB2, and SUB3 will be used. Each of these programs and the utility sorts will be run as separate activities.

SUB1 should be run first. It requires the input files THES and TEXT, and creates the scratch files (on tape, disc, or drum) ASSOC and OTHR as output. The ASSOC file will then be introduced as input for the sort activity. Records on this file will be exactly analogous to the ASCTA array records, written in binary mode, and the field on which to sort is the "Associate Word" field. (This corresponds to computer words 7 through 9 for each record.) The OTHR file will contain information from the other in-core arrays which it will be necessary to re-establish; it is also written in binary mode.

Both the sorted ASSOC file and the OTHR file will be used along with TEXT as input to the SUB2 program. This activity will produce a revised ASSOC file and OTHR file. This new ASSOC file must now be sorted once more, using location 4 (Backlink to the primary words) as the primary sorting field (numeric, in ascending order), and locations 7 through 9 (the Associate Word field) as the secondary sorting field.

The newly-sorted Assoc file and the OTHR file will be used as input to SUB3 along with the SYSIN parameters. SUB3 will call the subroutines

necessary to complete processing for this alternate version of THESR. All common regions, file formats, and parameter specifications will remain the same as for the standard version of THESR. A flowchart illustrating the proper sequence for execution appears on page 166.

IV. NOTES FOR PROGRAMMERS

SUBROUTINES:

The major processing subroutines are briefly described on the following pages:

DECODE:

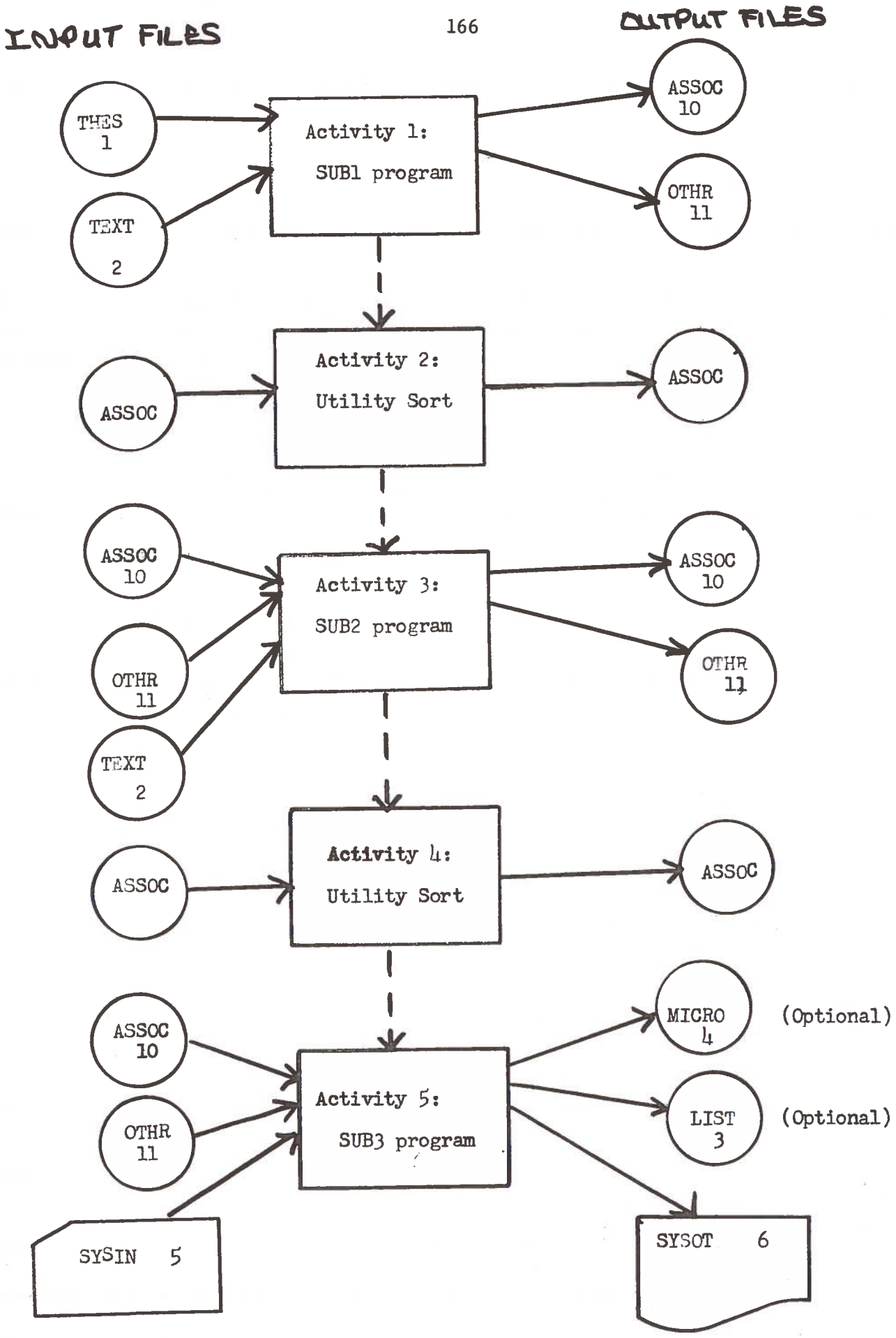
The DECODE subroutine reads input parameters from SYSIN and sets the flags in labeled common regions COMM and COMR2 to indicate what choices have been made.

THESR1

The THESR1 subroutine reads primary and associate words from a previously prepared thesaurus file (THES) and compares each primary word to a file of text words (TEXT--output from the SUFFIX program or the SELECT routine). When a match is found, the primary word is recorded in special array PRIMA; all words which are associated directly with it in THES are recorded in special array ASCTA. When a primary word is matched to a text word whose root is the same, but which appears in a slightly altered form, the alternate form is recorded in array ALTP.

THESR2

The THESR2 subroutine will access associate words and compare them to the file of text words. Those associate words which do not appear in TEXT will be dropped from processing by setting their match count field to zero. The remainder will be assigned the appropriate match counts and frequency counts, and the alternate forms which appear in TEXT will be recorded in array ALTA.


INPUT FILES
OUTPUT FILES
THES 1
TEXT 2
Activity 1: SUB1 program
ASSOC 10
OTHR 11
ASSOC
Activity 2: Utility Sort
ASSOC
ASSOC 10
OTHR 11
TEXT 2
Activity 3: SUB2 program
ASSOC 10
OTHR 11
ASSOC
Activity 4: Utility Sort
ASSOC
ASSOC 10
OTHR 11
Activity 5: SUB3 program
MICRO 4
(Optional)
LIST 3
(Optional)
SYSIN 5
SYSOT 6

THESR3

The THESR3 subroutine compares associate words to primary words. When a match is found, a pointer in the associate word record is set to indicate the appropriate primary word. This establishes a forward link for the THESR4 processing subroutine.

THESR4

The THESR4 subroutine builds and outputs the 5-level list structure of thesaurus words. It accesses the array of primary words and the array of associate words, as well as alternate forms for both primaries and associates, in a nested sequence. The list is output to SYSOT, and optionally to a file in binary mode on tape, disc, or drum which will be saved (LIST).

OUTP

The OUTP subroutine is called from THESR4 to output a record to SYSOT, and optionally to LIST.

MIKE

The subroutine named MIKE accesses the pruned list of thesaurus words and prepares a list of primary and associate word pairs on file MICRO. This is optional output, and the subroutine will not be called at all unless this file is to be written.

SORTIT

The SORTIT subroutine produces a keying array (Word 6 of the ASCTA array) which contains pointers ordering the associate words alphabetically. Parameters are passed through labeled common area ORDR. A system utility sort may be substituted for this subroutine in the manner described previously (SEE: SORT SUBSTITUTION PROCEDURE).

STEM

The STEM subroutine compares two words character by character and consults a list of possible suffixes and exceptions in order to determine whether the two words may be considered to have the same root. It will return the value of 1 in the variable SAME if the roots were found to match. Otherwise, the variable is set to zero. Entry point SFMTC1 initializes suffix and exception lists. The labeled common region ROOT contains the parameters for this subroutine.

SPECIAL SUBROUTINES:

A brief description of each special-purpose, machine-specific subroutine which is called by THESR routines will follow:

LCOMP(WORD1, WORD2)

LCOMP is a function subroutine which logically compares two words. All comparisons are in the standard collating sequence. The value returned is as follows:

-1	Word 1 is logically less than Word 2
0	Word 1 is equal to Word 2
1	Word 1 is logically greater than Word 2

COMPCT(INTO, IFROM, NCHAR)

COMPCT is a special routine which packs the first character from each of NCHAR words of array IFROM into array INTO at six characters per word. If NCHAR is not a multiple of six, the last word is blank filled on the right.

BRKUP(INTO,IFROM, NWDS)

BRKUP will unpack NWDS words of array IFROM and store them into array INTO at one character per word, left-justified with blank fill to the right.

NOTE: The routines LCOMP, COMPCT, and BRKUP are library routines for the Kansas University HW 635.

IBNRY(IWORD)

IBNRY is a function subroutine. IWORD is a one-digit number in character representation (A1) left-justified in the word. IBNRY converts this number to the normal integer representation, which becomes the value returned by the routine.

The FORTRAN version of IBNRY should be machine-independent, as it works on a table lookup basis. However, since it is a fairly simple routine and is called often, an assembly-language version which is tailored to the installation might improve overall program efficiency.

COMMON REGIONS:

The THESR subroutines communicate by means of labeled common regions which contain all of the important processing variables and arrays. These variables and arrays are therefore grouped according to the common regions to which they belong for description:

COMMON /ROOT/ IWORD(DIM), JWORD(DIM), SAME, NCHAR

This area is used primarily for communication with the STEM subroutine.

IWORD and JWORD contain characters stored at one per computer word, left-justified with blank fill to the right. These are the words which are to be compared by STEM. They are each dimensioned to the value of NCHAR. (DIM = NCHAR)

SAME assumes a value of zero if the words do not match, one if they are found to have the same root.

NCHAR has the value 18, which is the number of characters of interest for all processing done by THESR.

COMMON /ORDR/ ASCTA(D1M1, D1M2), ASCNT, KEY, KEYLEN

This area is used for communication with the sorting subroutine, and contains the array of associate words from the thesaurus.

KEY and KEYLEN are indicators for the sort field.

ASCNT is a count of the number of entries in the ASCTA array.

ASCTA: Each entry in this array is a record consisting of 6+NWDS computer words. This is specified by D1M1; for the HW635, D1M1=9. D1M2 should be large enough to accommodate the associate words which will be processed by THESR.

ASCTA array records:

Record Loc.	Contents
1	Match Count
2	Frequency
3	Alternates
4	Back Link ⟶ to Primary Word
5	Forward Link ⟶ to Primary Word
6	Index field for Sort
7-9	Associate Word (packed)

COMMON /PRM/ PRIMA(D1M1 D1M2), PRMCNT

PRMCNT contains the number of entries in the PRIMA array.

PRIMA: Each entry in this array is a record consisting of 6+NWDS computer words. This is specified by D1M1, and for the HW635, D1M1=9. D1M2 should be large enough to accommodate the primary words which will be processed by THESR.

Record Loc.	Contents
1	Match Count
2	Frequency
3	Alternates
4	Star
5	Link to associates
6	Unused at present
7-9	Primary word, packed format

Note: The 'Alternates' field for both primary and associate records is a code for the first location and the number of entries in the alternates array. It is obtained by the formula:

Alternates = (Location * INCR) + Count

COMMON /PRMALT/ ALTP(NWDS,IDIM), PLTCNT

The ALTP array contains records of length NWDS in which are stored the alternate forms for primary words. IDIM should be large enough to accommodate the words whihc will be processed by THESR for this array. PLTCNT is a count of the number of entries in the array.

COMMON /ASCALT/ ALTA(NWDS, IDIM), ALTCTA

The ASCALT common area is entirely analogous to the PRMALT common area, except that it contains the alternate forms of associate words.

COMMON /TXT/ RECORD(21), WORD(NCHAR)

The RECORD portion of this common area is used to store the first 21 computer words from the TEXT file records. The WORD portion of this area is used to store the next 18 computer words, which correspond to the 18 characters which are of greatest interest to THESR.

COMMON /ALTS/ ARRAY(NCHAR,9), ALTCNT, INCR, THREE(NWDS)

ARRAY is a temporary collection array for alternate forms of primary or associate words. The arbitrary limit of nine alternates for any one thesaurus word was chosen, but may be changed simply by altering the array dimension and the value of INCR. INCR should have the value of one greater than the maximum number of entries allowable in ARRAY. It is used to check for exceeding array bounds and for calculating the Alternates entry for the PRIMA and ASCTA arrays.

COMMON /WORK/ STG(NCHAR), SMAT, FREQ, STAR

Working storage areas:

STG is used to store NCHAR characters from time to time, usually for comparison purposes. SMAT is used to save the current match count, FREQ to save the frequency count, and STAR is used to indicate the value of the Star field in the primary word array.

COMMON /WORK2/ NWDS, IEND

NWDS is a variable containing the number of computer words which are required for storage of NCHAR characters when it is packed. For NCHAR=18, the HW635 will use 3 words of storage, 6 characters each. Therefore NWDS=3. IEND is a constant which is convenient. It is equal to NWDS+6.

COMMON /CONSTS/ Zero, IBlank

Self-explanatory: Zero and Iblank are constants.

COMMON /FILES/ THES,TEXT,MICRO,LIST,SYSIN,SYSOT

This is a communication region which transfers the values which have been assigned to the input and output files. THES=1, TEXT=2, MICRO=4, LIST=3, SYSIN=5, SYSOT=6.

COMMON /comm/ ISAVE, IDIM,IFMTS, MAT(2),LEVEL

Communication region which transfers processing information from the parameter cards. ISAVE indicates whether the LIST file is to be prepared, IFMTS indicates whether the Formats are to be read from SYSIN. LEVEL indicates the level to which the list is to be linked. IDIM and MAT are used when variable formats are to be read. IDIM indicates the maximum number of words to read from each card, and MAT is a variable containing the format specifications with which to read the variable formats.

COMMON /COMR2/ MICRO

This is a flag variable which indicates whether a text-specific thesaurus of word pairs is to be produced as output from THESR.

COMMON /FILES/ THES, TEXT, MICR, LIST, SYSIN, SYSOT

This is a region which contains the variable names of the input and output files. These files are assigned logical unit numbers only once, in the mainline, and consequently need only to be changed at that point since they are included with every subroutine.

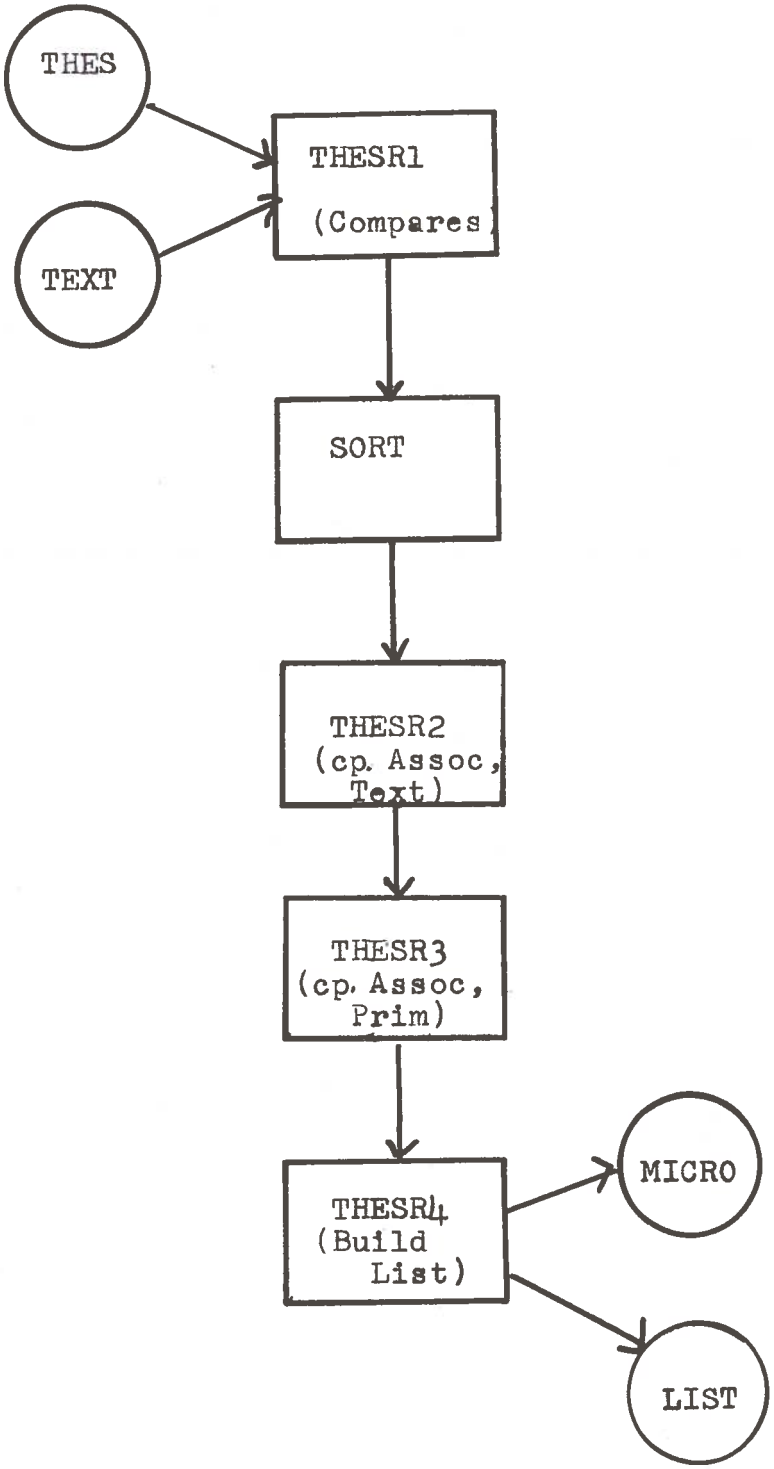
FURTHER NOTES FOR PROGRAMMERS:

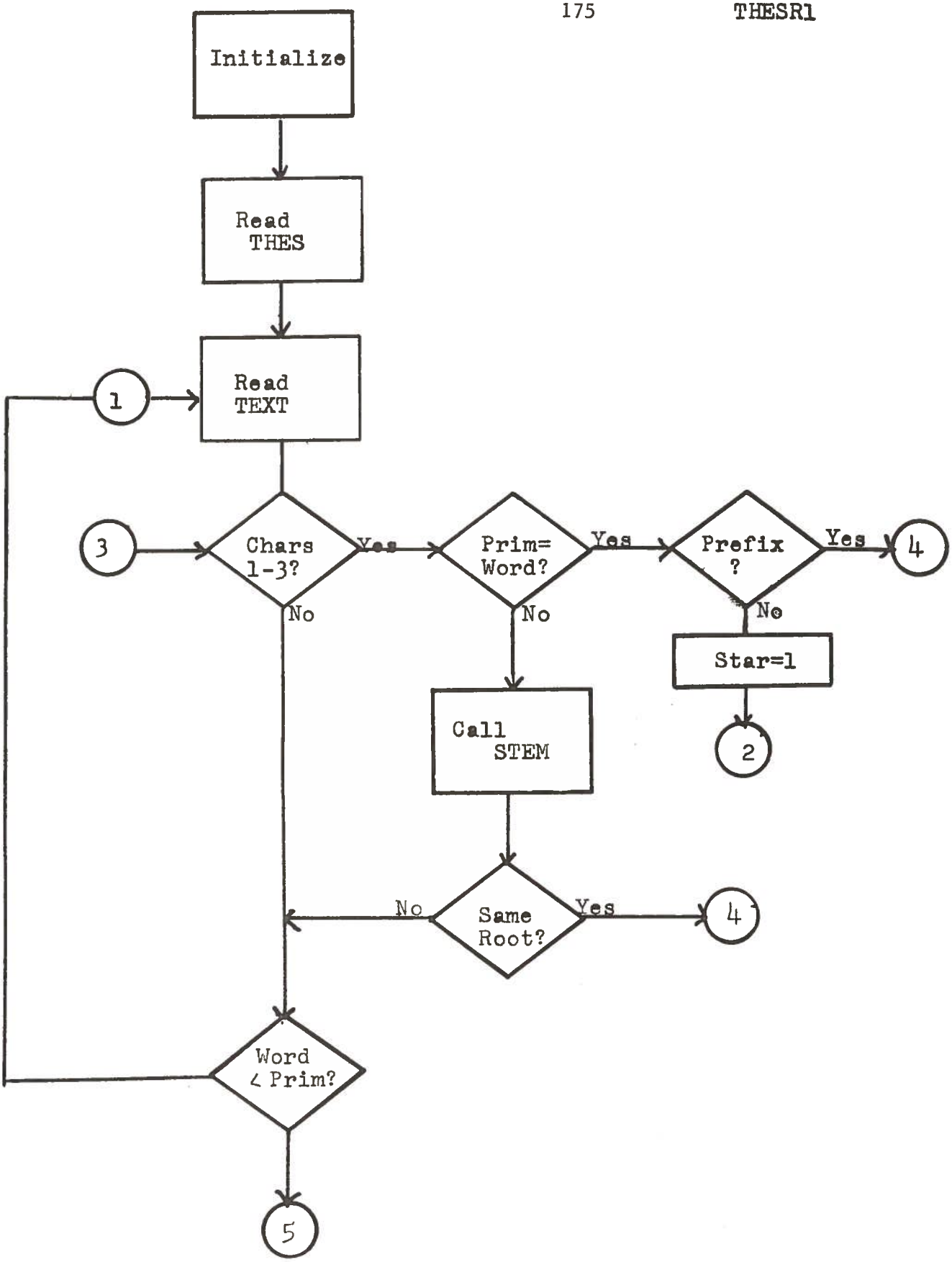
Nearly all of the variables and arrays which will be affected by conversion to another installation appear in the labeled common regions. However, each subroutine initialization section should be checked for compatibility if any variables in the common regions are changed, since there are some few dimension statements included.

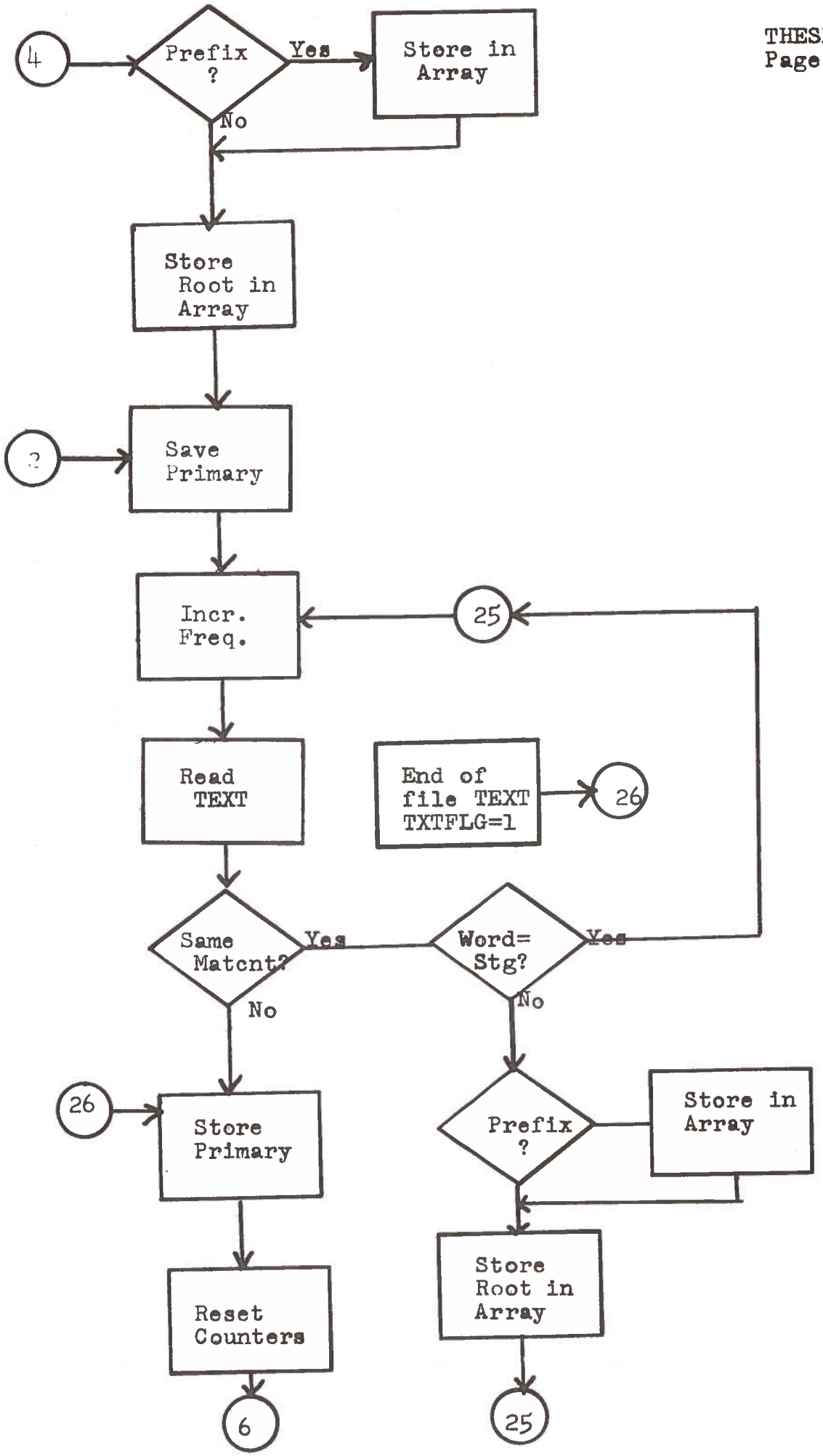
Since labeled common is not available at some installations, these areas should be combined into a blank common region and used for all subroutines, noting carefully the differences in variable names which sometimes occur between subroutines.

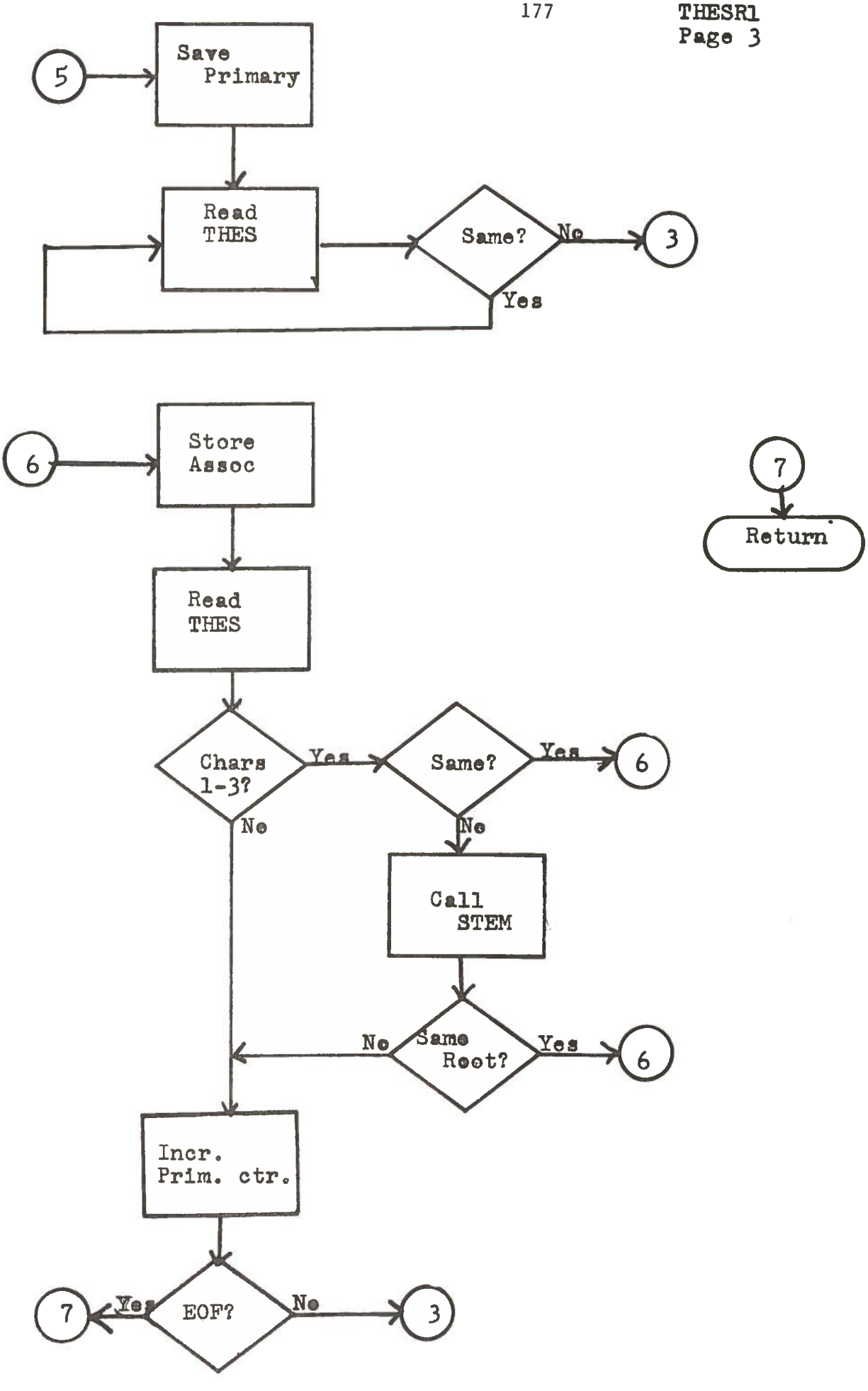
Values which must be assigned to variables and do not change (constants and file logical unit numbers, for example) are initialized in the mainline, and hence need to be changed only at that point, if at all. A block data subprogram is included in order to assign Hollerith constants to variables in common regions.

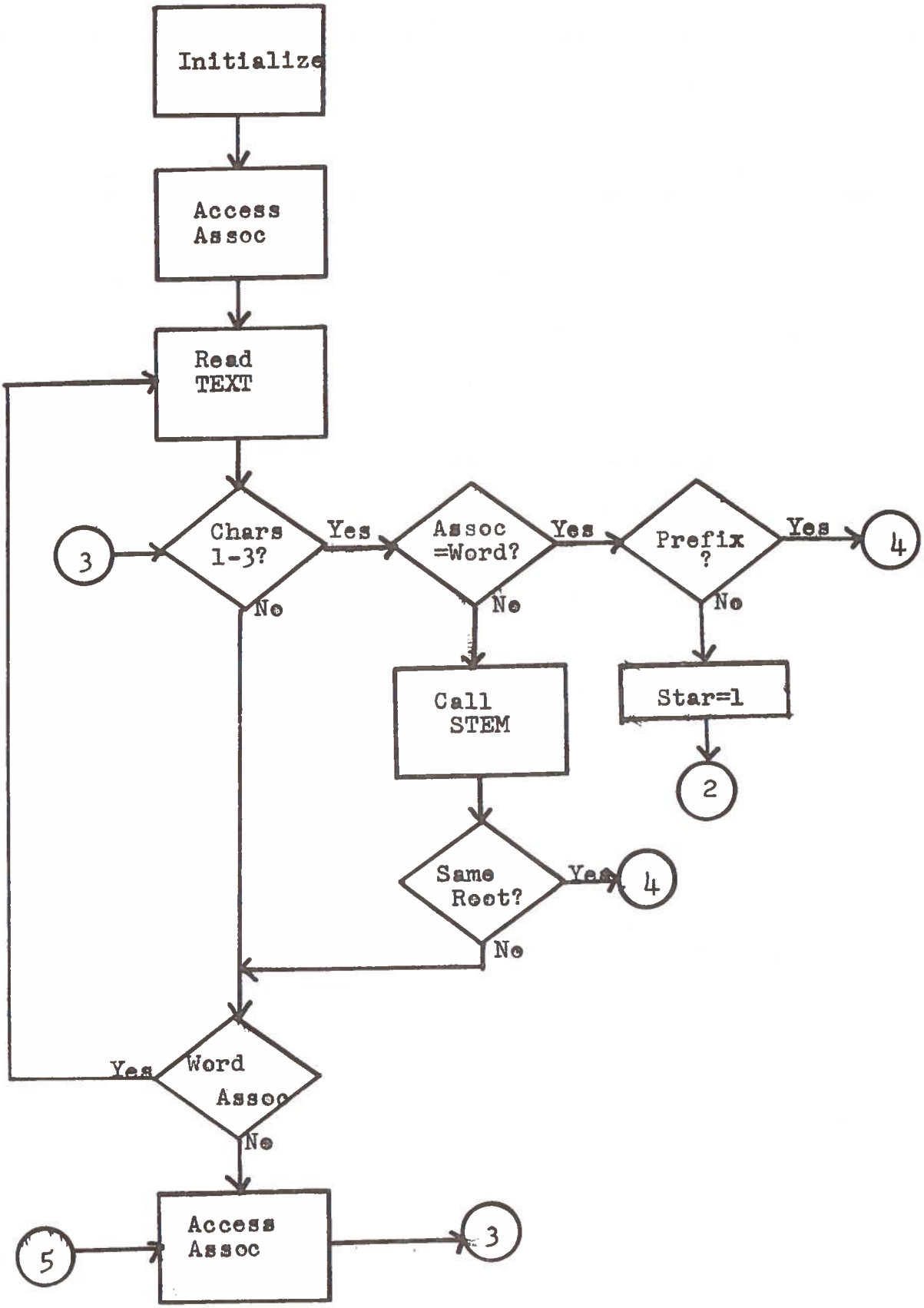
The PRIMA and ASCTA arrays have been designed similarly so that they may be combined into one array and accessed from opposite ends to achieve a crude dynamic allocation scheme. The ALTP and ALTA arrays are also arranged in this manner.

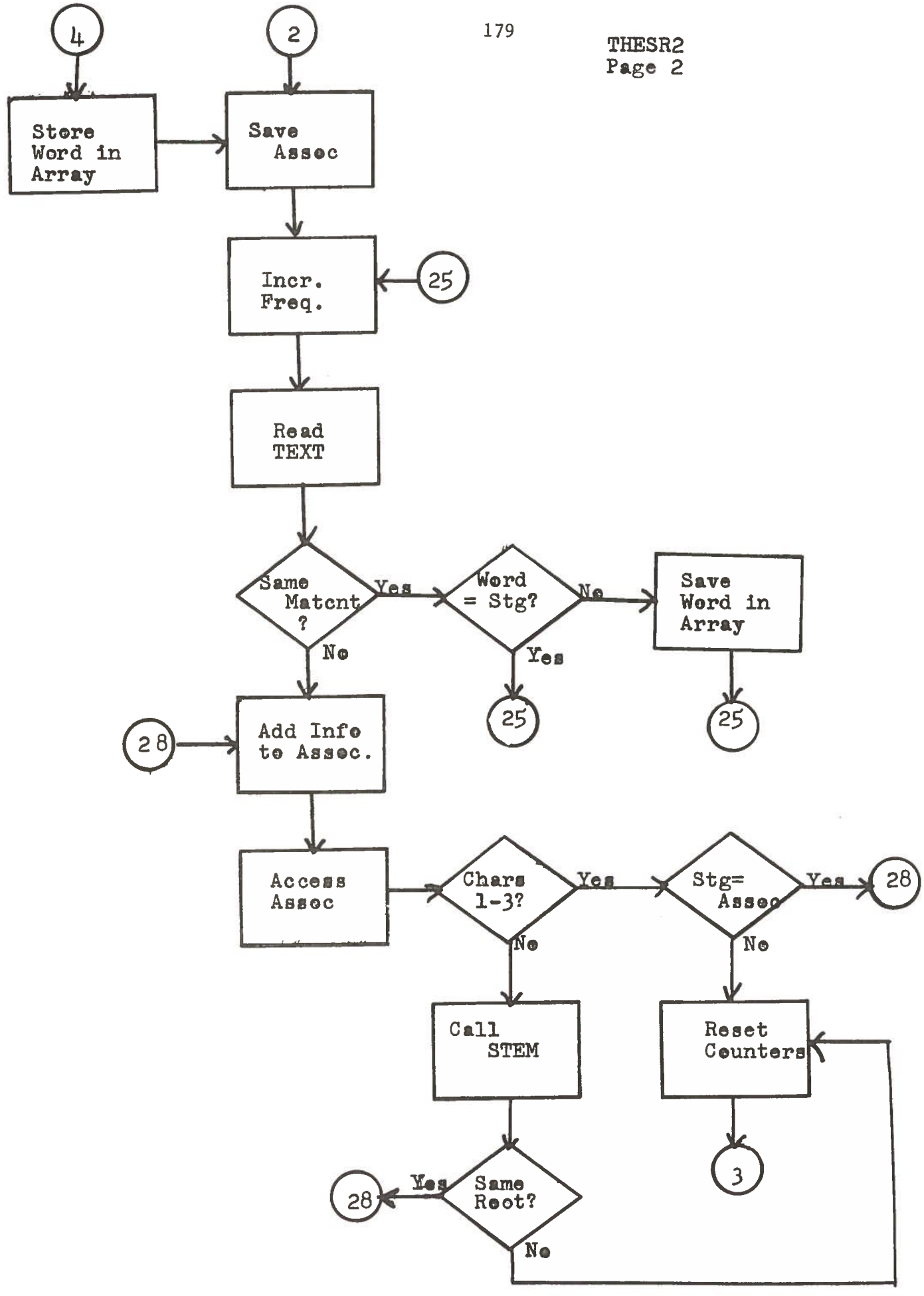

THES
TEXT
THESR1
(Compares)
SORT
THESR2
(cp. Assoc,
Text)
THESR3
(cp. Assoc,
Prim)
THESR4
(Build
List)
MICRO
LIST

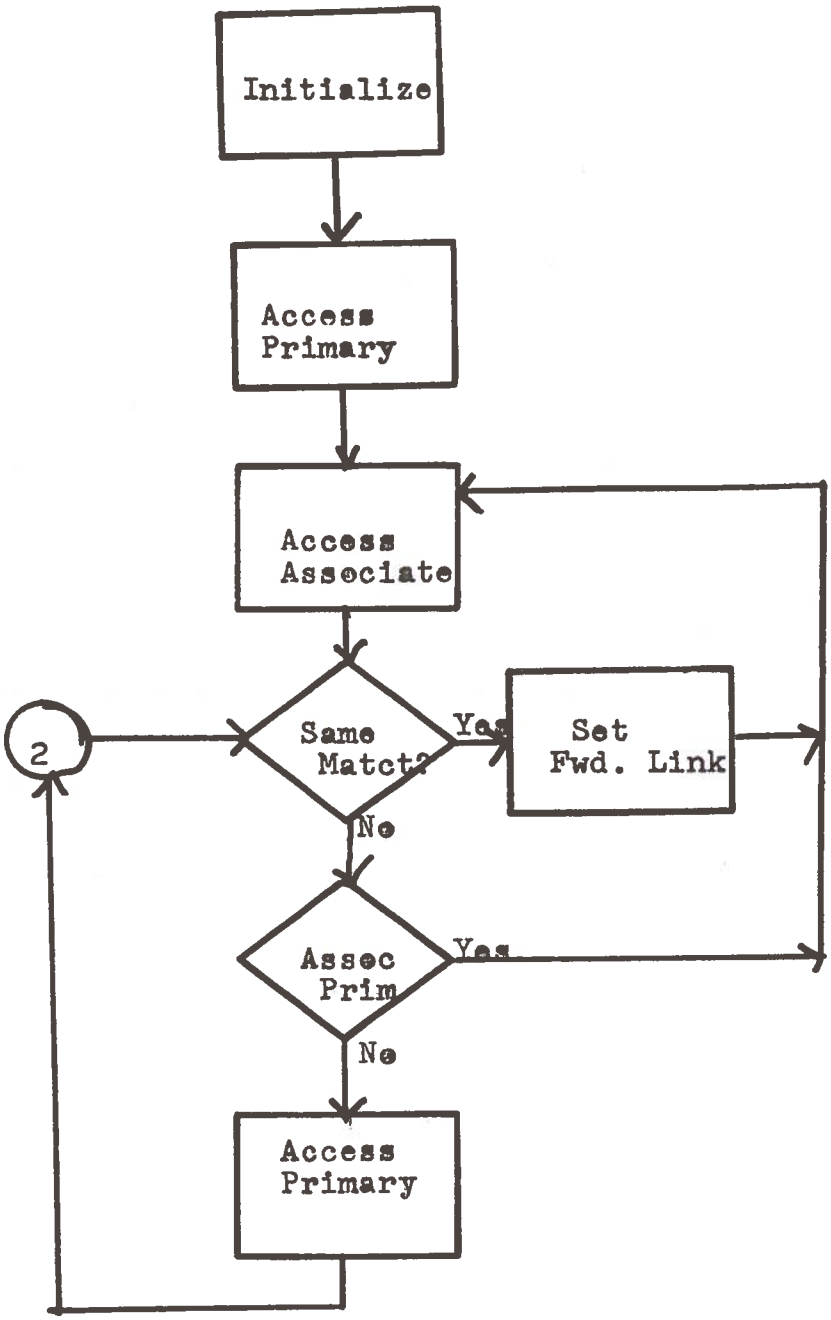

Initialize
Read THES
Read TEXT
1
3
Chars 1-3?
Yes
No
Prim= Word?
Yes
No
Prefix ?
Yes
No
4
Star=1
2
Call STEM
Same Root?
No
Yes
4
Word < Prim?
5

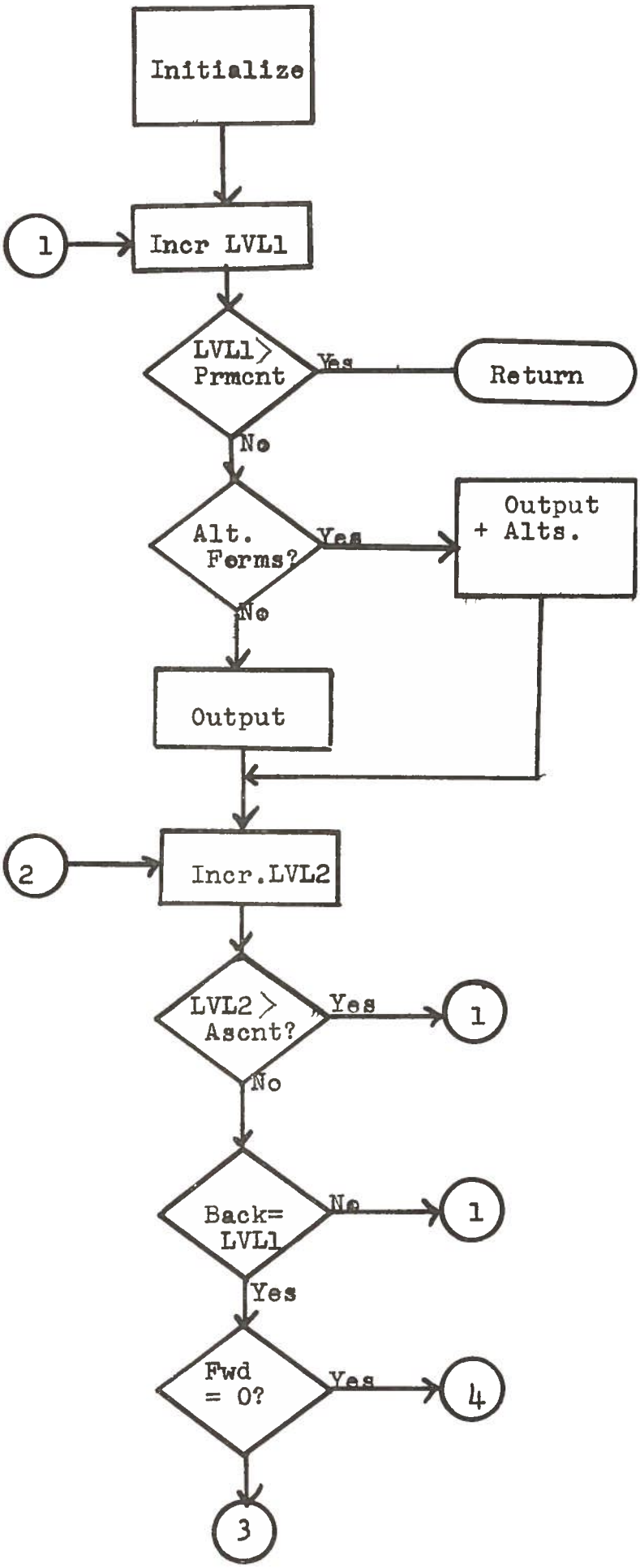

4
Prefix ?
Yes
Store in Array
No
Store Root in Array
2
Save Primary
Incr. Freq.
25
Read TEXT
End of file TEXT TXTFLG=1
26
Same Matcnt?
Yes
Word= Stg?
Yes
No
No
26
Store Primary
Prefix ?
Store in Array
Reset Counters
Store Root in Array
6
25

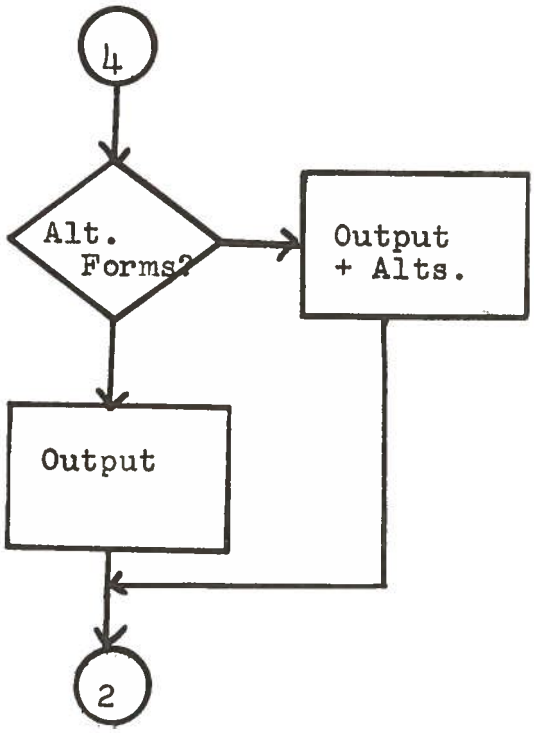

5
Save Primary
Read THES
Same?
No
3
Yes
6
Store Assoc
Read THES
Chars 1-3?
Yes
Same?
Yes
6
No
No
Call STEM
No
Same Root?
Yes
6
Incr. Prim. ctr.
7
Yes
EOF?
No
3
7
Return

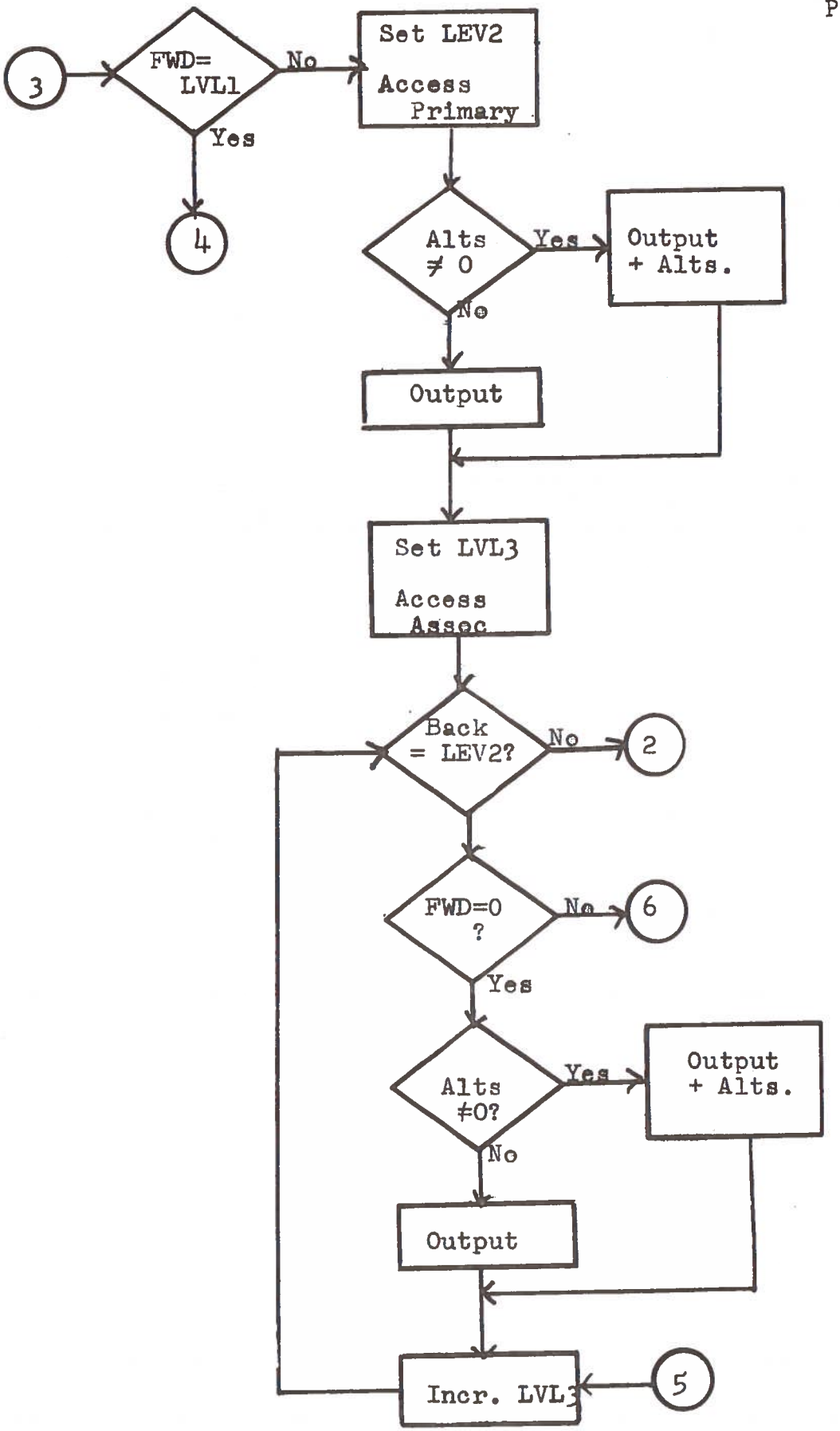

Initialize
Access Assoc
Read TEXT
3
Chars 1-3?
Yes
No
Assoc =Word?
Yes
No
Prefix ?
Yes
No
4
Call STEM
Star=1
2
Same Root?
Yes
4
No
Word Assoc
Yes
No
5
Access Assoc
3

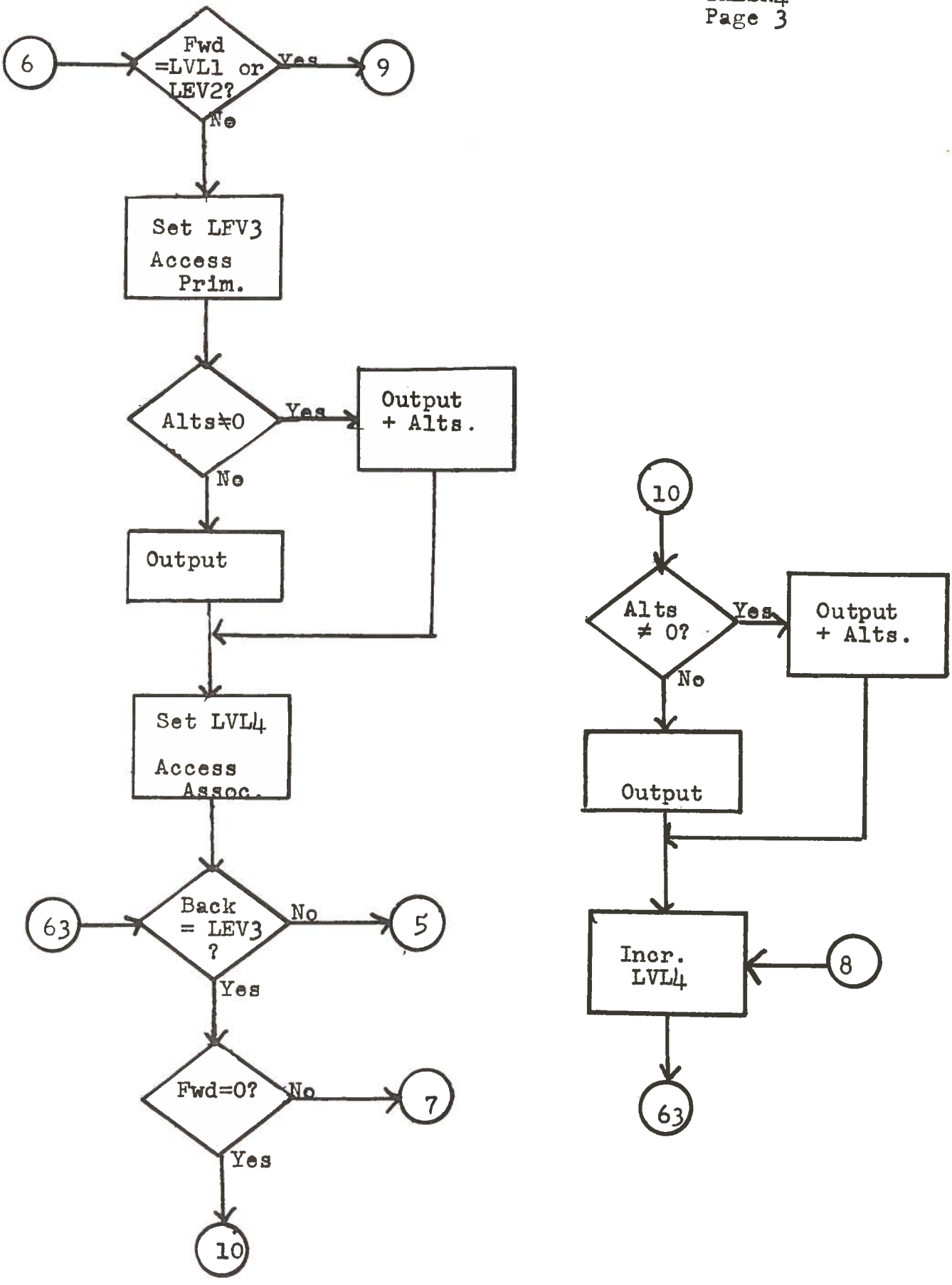

4
2
Store Word in Array
Save Assoc
Incr. Freq.
25
Read TEXT
Same Matcnt ?
Yes
Word = Stg?
No
Save Word in Array
No
Yes
25
25
28
Add Info to Assoc.
Access Assoc
Chars 1-3?
Yes
Stg= Assoc
Yes
28
No
No
Call STEM
Reset Counters
3
28
Yes
Same Root?
No

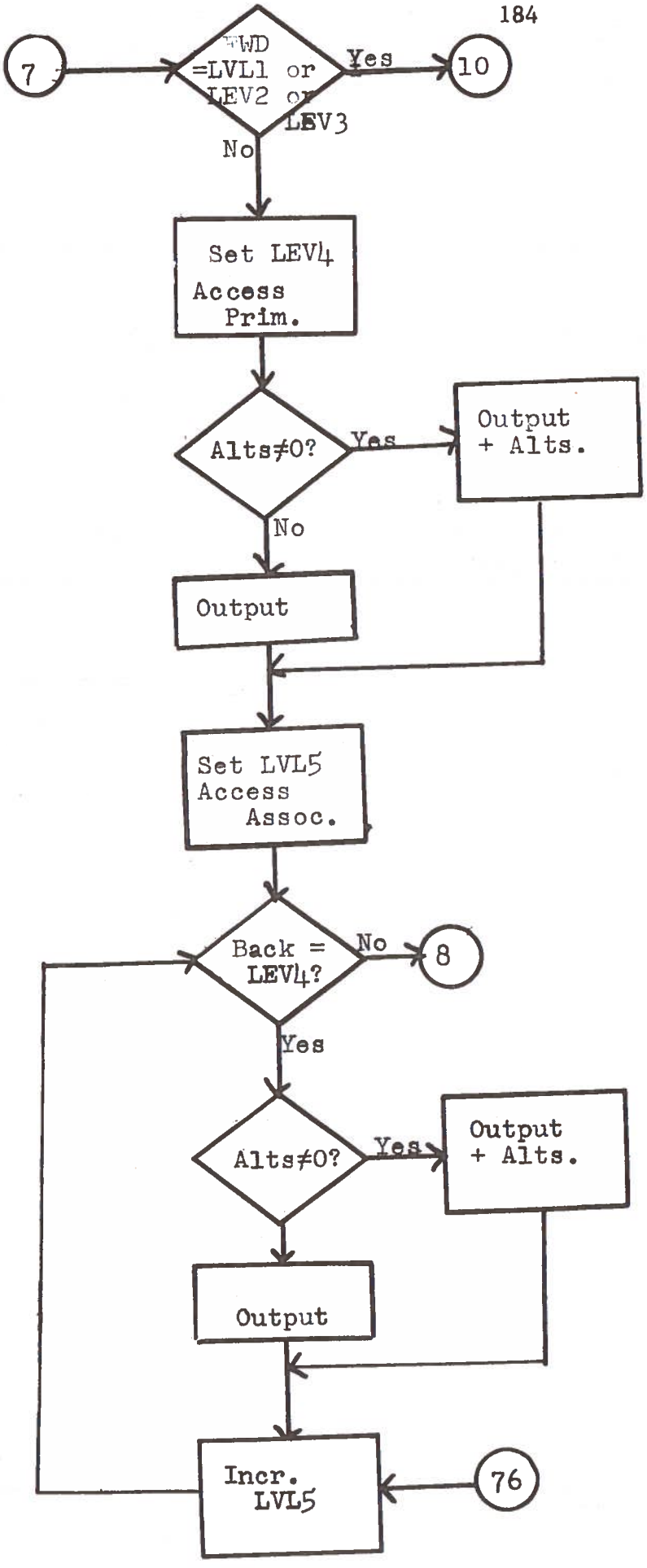

Initialize
Access Primary
Access Associate
Same Matct?
Yes
Set Fwd. Link
No
Assec Prim
Yes
No
Access Primary
2


Initialize
1
Incr LVL1
LVL1 > Prmcnt
Yes
Return
No
Alt. Forms?
Yes
Output + Alts.
No
Output
2
Incr.LVL2
LVL2 > Ascnt?
Yes
1
No
Back= LVL1
No
1
Yes
Fwd = 0?
Yes
4
3


4
Alt. Forms?
Output + Alts.
Output
2


3
FWD= LVL1
No
Yes
Set LEV2
Access Primary
4
Alts ≠ 0
Yes
Output + Alts.
No
Output
Set LVL3
Access Assoc
Back = LEV2?
No
2
FWD=0 ?
No
6
Yes
Alts ≠0?
Yes
Output + Alts.
No
Output
Incr. LVL3
5


6
Fwd =LVL1 or LEV2?
Yes
9
No
Set LFV3 Access Prim.
Alts≠0
Yes
Output + Alts.
No
Output
Set LVL4 Access Assoc.
63
Back = LEV3 ?
No
5
Yes
Fwd=0?
No
7
Yes
10
10
Alts ≠ 0?
Yes
Output + Alts.
No
Output
Incr. LVL4
8
63


7
TWD =LVL1 or LEV2 or LEV3
Yes
10
No
Set LEV4 Access Prim.
Alts≠0?
Yes
Output + Alts.
No
Output
Set LVL5 Access Assoc.
Back = LEV4?
No
8
Yes
Alts≠0?
Yes
Output + Alts.
Output
Incr. LVL5
76

III. Professional Activities of Project Personnel

Sally Y. Sedelow

Publications:

"Computers and Language," Iowa Alumni Review, June-July, 1971

Automated Analysis of Language Style and Structure, Report on research for the period March 1, 1970 to August 31, 1971, Contract N000 14-67-A-0321-001, Office of Naval Research, University of North Carolina. DDC # AD 711-643. 162 pp.

Papers/Seminars/Addresses/etc.:

"Models, Computing and Stylistics," Mount Holyoke College, Sigma Xi Lecture, December, 1970.

Lecture on Computer-Aided Stylistic Analysis: Skidmore College, April, 1971.

Lecture on Computer-Aided Stylistic Analysis: Southeastern Modern Language Association, Annual Meeting, Movember, 1970.

Lecture on Computer-Aided Analysis of Language: Colloquium, Linguistics and Computer Science, University of Kansas, 1971.

Activities:

Co-Editor, Computer Studies in the Humanities and Verbal Behavior, 1966--.

Principal Investigator, NSF Study re Possible National Center/ Network for Computational Research on Language, 1971-72.

Committee on Information Technology, American Council of Learned Societies, 1970--.

Field Reader of Proposals, U. S. Department of Health, Education, and Welfare, 1966--.

Proposal Evaluation, Canada Council, 1968--.

Proposal Evaluation, National Endowment for the Humanities, 1969--.

Proposal Evaluation, Special Projects Program, NSF, 1970--.

Reviewer of Papers for FJCC, 1968-- and SJCC, 1969--.

Chairman, Investigative Committee on "Shakespeare and the Computer," for World Shakespeare Congress, August, 1971, Vancouver.

Member, Technical Area Committee 7 on "Science and Humanities" for Fifth IFIP Congress, Ljubljana, Yugoslavia, 1971.

Reviewer of Papers, IFIP Congress, 1971.

Reviewer of Papers, ACM, 1971.

Reviewer and Session Chairman, Humanities, Conference on Computers in the Undergraduate Curriculum, Dartmouth, 1971.

Instructor, NSF Summer Workshop for Faculties of Lynchburg, Randolph-Macon and Sweet Briar Colleges, 1971.

Site Visitor, COSIP Program, 1971.

Session Chairman, "Time-Shared Text and Information Handling," FJCC, 1970.

Walter A. Sedelow, Jr.

Publications:

Review: Harold Pepinsky, ed., "People and Information," Social Forces, (In Press).

Papers/Seminars/Addresses/etc.:

"Social Implications of Computing," 1971 GE Users Meeting (GESHUA XII).

"The Liberal Sciences Curriculum and Computer Science," lectures for a special National Science Foundation-supported institute for faculty at Sweet Briar College-Randolph Macon Women's College-Lynchburg College, June, 1971.

"Some Prospects and Ramifications for Computational Research on Language," Computer Science Colloquium, University of Kansas, February, 1971.

Address at installation of new members, Phi Alpha Theta (History honorary fraternity), University of Kansas Chapter, 1971.

Activities:

Series Editor, The Free Press/MacMillan Company, New York.

Journal Board of Editors, for Sociology, Computer Studies in the Humanities and Verbal Behavior.

Referee, Social Forces.

Referee, Office of Science Information Service, National Science Foundation.

Referee, IFIP (International Federation for Information Processing) Congress '71.

Referee, National Endowment for the Humanities.

Consultant, Automated Analysis of Language Style and Structure in Technical and Other Documents, sponsored by the U. S. Office of Naval Research at KU.

Research under a KU General Research Fund grant on preliminary studies in the quantified analysis of the language of sociology.

Principal Investigator, Study Grant re a possible national center/network for computational research on language (CeNCoReL), National Science Foundation.

Study for a 'liberal sciences' undergraduate curriculum.

Peggy Lewis

Activities:

1970-71 Chairman of the Kansas University Student Chapter of the Association for Computing Machinery.

Sam Warfel

Activities:

1970-71 Award as the outstanding graduate student in Linguistics.

Martin Dillon

Publications:

"An Approach to Matrix Generation and Report Writing for a Class of Large-Scale Linear Programming Models," Applications of Mathematical Programming Techniques, pp. 171-184, London, 1970. Co-author with P. Michael Jenkins and Mary J. O'Brian.

"The Structure and Content of Scientific Discourse," Research Previews (November, 1970), pp. 1-5.

"Preliminary Design Considerations of a Language Analysis Support Package," in Automated Analysis of Language Style and Structure, 1969-1970, Chapel Hill: University of North Carolina, pp. 18-37.

Principles of Operations Research by Harvey M. Wagner (Englewood Cliffs, N.J.: Prentice-Hall, 1969) in Information, Storage and Retrieval, Vol. 7, pp. 147-148.

Style and Vocabulary: Numerical Studies, by C.B. Williams (New York: Hafner Publishing Co., 1970) in Technometrics, Vol. 13, No. 3, pp. 708-709.

Papers/Seminars/Addresses/etc.:

"Advanced Basis Selection for a Class of Linear Programming Models," Proceedings of the Ninth Annual Regional Conference of the ACM, May, 1970.

Activities:

Principal Investigator, "Automated Analysis of Interdisciplinary Discourse Barriers," funded by National Aeronautics and Space Administration.

Principal Investigator, "Analysis of Automated Information Systems for Population Planning," funded by Agency for International Development.

Co-Investigator, "Information, Values and Urban Policy Formation," funded by National Institute of Mental Health.

David Jon Wagner

Publications:

"Thesaurus Research" in Sally Yeates Sedelow, Automated Analysis of Language, Style and Structure 1969-1970.

"User's Manual for VIA" in Sally Yeates Sedelow, Automated Analysis of Language, Style and Structure 1969-1970.

Activities:

Consultant, Frank Porter Graham Child Development Center

Consultant, Institute for Research in the Social Sciences at the University of North Carolina

Consultant, Department of Marine Sciences, The University of North Carolina

Member, Graduate Faculty of the University of North Carolina

Trainee, National Science Foundation

Ph.D. Candidate, The University of North Carolina

IV. APPENDIX

```
CMAIN       MAINLINE                                                          10
C                                                                             20
C       SUBROUTINES THAT ARE SOFTWARE AND MACHINE DEPENDENT --                30
C       CLOCK1                                                                40
C       FLGEOF                                                                50
C                                                                             60
C                                                                             70
C       CORRESPONDENCE OF COUNTERS FOR VARIOUS PROCESSING MODES -             80
C        -PROS-     -MILT-     -PLAY-     -VPLA-     -POET-     -SPOK-        90
C          VOLUME     VOLUME     ACT        ACT        1          SERIES     100
C          CHAPTER    BOOK       SCENE      SCENE      1          SESSION    110
C          PARAGRAPH  PARAGRAPH  PARAGRAPH  PARAGRAPH  STANZA     SPEAKER    120
C          SENTENCE   LINE       SENTENCE   LINE       LINE       SENTENCE   130
C          WORD       WORD       WORD       WORD       WORD       WORD       140
C                                                                            150
C                                                                            160
      INTEGER COMMA,BLANKS                                                   170
      INTEGER PARAMS(80)                                                     180
      INTEGER AMPERS                                                         190
C-----------------------------------------------------------------------     200
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.              210
      INTEGER EOFLAG                                                         220
      LOGICAL READF                                                          230
      LOGICAL FIRSTF                                                         240
      LOGICAL PRINTF,SEQFLA,STAGEF                                           250
      INTEGER CONTIN                                                         260
      INTEGER WORDS(25,6)                                                    270
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       280
      INTEGER TYPEPR                                                         290
      INTEGER CARDCN,OUTCNT                                                  300
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     310
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            320
      INTEGER WRDPTR                                                         330
      COMMON /SYMBOL/ WORDS                                                  340
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    350
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         360
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        370
      COMMON PRINTF,SEQFLA,STAGEF                                            380
      COMMON FIRSTF                                                          390
      COMMON READF                                                           400
      COMMON EOFLAG                                                          410
C-----------------------------------------------------------------------     420
C                                                                            430
      EQUIVALENCE (COMMA,WORDS(9,3))                                         440
      EQUIVALENCE (BLANKS,WORDS(9,6))                                        450
      EQUIVALENCE (AMPERS,WORDS(7,6))                                        460
C                                                                            470
      DATA MASK77/O7777777777777/                                            480
C                                                                            490
C                                                                            500
C     INITIALIZE BLANK COMMON VARIABLES.                                     510
      CALL INITCM                                                            520
C                                                                            530
C     INITIALIZE EOFLAG.                                                     540
C     EOFLAG - 0 OK, 1 EOF CONDITION READ.                                   550
      CALL FLGEOF(INFILE,EOFLAG)                                             560
C                                                                            570
C     GET TODAYS DATE.                                                       580
      CALL CLOCK1(IDAY,IMON,IYR)                                             590
```

```
      READ(SYSIN,9002) PARAMS                                          600
9002  FORMAT(80A1)                                                     610
      READ(SYSIN,9000) TITLE                                           620
9000  FORMAT(20A4)                                                     630
C     INSERT COMMA AT END OF PARM STRING.                              640
460   IPTR=81                                                          650
461   IPTR=IPTR-1                                                      660
      IF(IPTR) 1420,1420,462                                           670
462   IF(PARAMS(IPTR) .EQ. BLANKS) GO TO 461                           680
      PARAMS(IPTR+1)=COMMA                                             690
C     SCAN DOWN TO FIRST NONBLANK COLUMN.                              700
      DO 463 J1=1,80                                                   710
      IF(PARAMS(J1) .NE. BLANKS) GO TO 464                             720
463   CONTINUE                                                         730
      GO TO 1420                                                       740
C     IPTR POINTS TO LAST COLUMN LOOKED AT.                            750
464   IPTR=J1-1                                                        760
C     IS FIRST PARM ,PROC=, ---                                        770
469   DO 470 J1=1,5                                                    780
      L=J1+IPTR                                                        790
      IF(PARAMS(L) .NE. WORDS(1,J1)) GO TO 590                         800
470   CONTINUE                                                         810
C     FIND OUT KEYWORD - (PROS,POET,PLAY,VPLA,MILT,SPOK).              820
C             TYPEPR --   (2,3,4,5,6,7)                                830
      DO 540 ITYPE=2,7                                                 840
      DO 541 J1=1,5                                                    850
      L=J1+IPTR                                                        860
      IF(PARAMS(L+5) .NE. WORDS( ITYPE,J1)) GO TO 540                  870
541   CONTINUE                                                         880
C     REMEMBER TYPE OF PROCESSING.                                     890
      TYPEPR=ITYPE                                                     900
      GO TO 560                                                        910
540   CONTINUE                                                         920
      GO TO 990                                                        930
560   J=10                                                             940
      GO TO 1050                                                       950
C                                                                      960
C     IS NEXT PARM ,PRINT=, ---                                        970
590   DO 591 J1=1,6                                                    980
      L=J1+IPTR                                                        990
      IF(PARAMS(L ) .NE. WORDS(8,J1)) GO TO 720                       1000
591   CONTINUE                                                        1010
C     IS PRINT KEYWORD ,NO,, ---                                      1020
      DO 600 J1=1,3                                                   1030
      L=J1+IPTR                                                       1040
      IF(PARAMS(L +6) .NE. WORDS(9,J1)) GO TO 650                     1050
600   CONTINUE                                                        1060
      PRINTF=.FALSE.                                                  1070
      J=9                                                             1080
      GO TO 1050                                                      1090
C     IS PRINT KEYWORD ,YES,, ---                                     1100
650   DO 651 J1=1,4                                                  1110
      L=J1+IPTR                                                       1120
      IF(PARAMS(L +6) .NE. WORDS(10,J1)) GO TO 990                    1130
651   CONTINUE                                                        1140
      PRINTF=.TRUE.                                                   1150
      J=10                                                            1160
      GO TO 1050                                                      1170
C                                                                     1180
C     IS NEXT PARM ,SEQ=, ---                                         1190
```

```
720     DO 721 J1=1,4                                                   1200
        L=J1+IPTR                                                       1210
        IF(PARAMS(L ) .NE. WORDS(11,J1)) GO TO 850                      1220
721     CONTINUE                                                        1230
C       IS SEQ KEYWORD ,NO,, ---                                        1240
        DO 730 J1=1,3                                                   1250
        L=J1+IPTR                                                       1260
        IF(PARAMS(L +4) .NE. WORDS(9,J1)) GO TO 780                     1270
730     CONTINUE                                                        1280
        SEQFLA=.FALSE.                                                  1290
        J=7                                                             1300
        GO TO 1050                                                      1310
C       IS SEQ KEYWORD ,YES,, ---                                       1320
780     DO 781 J1=1,4                                                   1330
        L=J1+IPTR                                                       1340
        IF(PARAMS(L +4) .NE. WORDS(10,J1)) GO TO 990                    1350
781     CONTINUE                                                        1360
        SEQFLA=.TRUE.                                                   1370
        J=8                                                             1380
        GO TO 1050                                                      1390
C                                                                       1400
C       IS NEXT PARM ,STAGE=, ---                                       1410
850     DO 851 J1=1,6                                                   1420
        L=J1+IPTR                                                       1430
        IF(PARAMS(L ) .NE. WORDS(12,J1)) GO TO 870                      1440
851     CONTINUE                                                        1450
C       IS STAGE KEYWORD ,NO,, ---                                      1460
        DO 860 J1=1,3                                                   1470
        L=J1+IPTR                                                       1480
        IF(PARAMS(L +6) .NE. WORDS(9,J1)) GO TO 910                     1490
860     CONTINUE                                                        1500
        STAGEF=.FALSE.                                                  1510
        J=9                                                             1520
        GO TO 1050                                                      1530
C       IS STAGE KEYWORD ,YES,, ---                                     1540
910     DO 911 J1=1,4                                                   1550
        L=J1+IPTR                                                       1560
        IF(PARAMS(L +6) .NE. WORDS(10,J1)) GO TO 990                    1570
911     CONTINUE                                                        1580
        STAGEF=.TRUE.                                                   1590
        J=10                                                            1600
        GO TO 1050                                                      1610
C                                                                       1620
C       IS NEXT PARM ,DELIM=, ---                                       1630
870     DO 871 J1=1,6                                                   1640
        L=J1+IPTR                                                       1650
        IF(PARAMS(L) .NE. WORDS(25,J1)) GO TO 990                       1660
871     CONTINUE                                                        1670
C       SUBSTITUTE DELIMITER.                                           1680
        AMPERS=PARAMS(L+1)                                              1690
        J=8                                                             1700
        GO TO 1050                                                      1710
C                                                                       1720
C       ERROR ROUTINE.                                                  1730
990     LIM1=IPTR+1                                                     1740
C       SEARCH FOR COMMA AT END OF PARM IN ERROR.                       1750
        DO 991 J=LIM1,80                                                1760
        IF(PARAMS(J) .EQ. COMMA) GO TO 992                              1770
991     CONTINUE                                                        1780
        GO TO 1420                                                      1790
```

```
992     LIM2=J                                                          1600
C                                                                       1610
        IF(FIRSTF) CALL ENDPAG                                          1620
        WRITE(SYSPRT,9001) (PARAMS(J1),J1=LIM1,LIM2)                    1630
9001    FORMAT(//,1H ,8HKEYWORD ,     1X,19HINVALID OR IN ERROR,        1640
     * 3H - ,80A1)                                                     1650
        IPTR=J                                                          1660
        GO TO 1051                                                      1670
C                                                                       1680
C       BUMP PAST KEYWORD WE JUST FOUND.                                1690
1050    IPTR=IPTR+J                                                     1700
C       ARE THERE MORE PARMS ---                                        1710
1051    IF(PARAMS(IPTR+1) .NE. BLANKS) GO TO 469                        1720
1070    GO TO (1420,1080,1110,1230,1140,1170,1200),TYPEPR               1730
C                                                                       1740
1080    CALL PRINT(4HPROS)                                              1750
        CALL PROS                                                       1760
        GO TO 1480                                                      1770
C                                                                       1780
1110    CALL PRINT(4HPOET)                                              1790
        CALL POET                                                       1800
        GO TO 1480                                                      1810
C                                                                       1820
1140    CALL PRINT(4HVPLA)                                              1830
        CALL PLAY                                                       1840
        GO TO 1480                                                      1850
C                                                                       1860
1170    CALL PRINT(4HMILT)                                              1870
        CALL MILT                                                       1880
        GO TO 1480                                                      1890
C                                                                       1900
1200    CALL PRINT(4HSPOK)                                              1910
        CALL SPOK                                                       1920
        GO TO 1480                                                      1930
C                                                                       1940
1230    CALL PRINT(4HPLAY)                                              1950
        CALL PLAY                                                       1960
        GO TO 1480                                                      1970
C                                                                       1980
C       ERROR ROUTINE.                                                  1990
1420    IF(FIRSTF) CALL ENDPAG                                          2000
        WRITE(SYSPRT,9020)                                              2010
9020    FORMAT(//,1H ,46HTYPE OF PROCESSING WAS NOT SPECIFIED OR WAS IN,2020
     * 7H ERROR.,/,1H ,32HJOB TERMINATED AT THIS POINT ***,//)          2030
C                                                                       2040
C       PUT OUT EOF INDICATION ON ,OUTFIL,.                             2050
1480    WORDN=0                                                         2060
C       THIS CHARACTER WILL ALWAYS SORT TO THE END OF FILE.             2070
        DO 998 J1=1,18                                                  2080
998     WORD(J1)=MASK77                                                 2090
        WRDPTR=19                                                       2100
        CALL WRTWRD                                                     2110
        END FILE OUTFIL                                                 2120
        REWIND OUTFIL                                                   2130
        REWIND INFILE                                                   2140
C                                                                       2150
        CALL ENDPAG                                                     2160
        WRITE(SYSPRT,9021) CARDCN,OUTCNT                                2170
9021    FORMAT(//,1H ,I6,5X,11HCARDS INPUT,//,1H ,I6,5X,                2180
     * 14HRECORDS OUTPUT,//,30X,22H***** END OF JOB *****)             2190
```

```
C
      CALL EXIT                                                         2400
      END                                                               2410
                                                                        2420
```

```
CWRTWRD     SUBROUTINE TO WRITE INDEXED WORD.                                 10
      SUBROUTINE WRTWRD                                                       20
C                                                                             30
C                                                                             40
C     OUTPUT RECORD FORMAT -                                                  50
C       WORD       LENGTH    FORMAT     FIELD                                60
C       1          1         I          LINEAR WORD NUMBER IN TEXT           70
C       2          1         I          VOLUME                               80
C       3          1         I          CHAPTER                              90
C       4          1         I          PARAGRAPH                           100
C       5          1         I          SENTENCE                            110
C       6          1         I          WORD-IN-SENTENCE                    120
C       7          1         I          PAGE NUMBER                         130
C       8          1         I          IDIOM FLAG                          140
C       9          1         I          PREFIX LENGTH IN CHARACTERS         150
C       10         1         I          WORD LENGTH IN CHARACTERS           160
C       11-18      8         A1         PREFIX CHARACTERS                   170
C       19-36      18        A1         TEXT WORD CHARACTERS                180
C                                                                             190
C                                                                             200
      INTEGER BLANKS                                                          210
      INTEGER PREFL                                                           220
      INTEGER WIS,SENTEN,PARAGR,CHAPTE,VOLUME                                 230
C-------------------------------------------------------------------------    240
      INTEGER EOFLAG                                                          250
      LOGICAL READF                                                           260
      LOGICAL FIRSTF                                                          270
      LOGICAL PRINTF,SEQFLA,STAGEF                                            280
      INTEGER CONTIN                                                          290
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.               300
      INTEGER WORDS(25,6)                                                     310
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                        320
      INTEGER TYPEPR                                                          330
      INTEGER CARDCN,OUTCNT                                                   340
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                      350
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                             360
      INTEGER WRDPTR                                                          370
      COMMON /SYMBOL/ WORDS                                                   380
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                     390
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN          400
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG         410
      COMMON PRINTF,SEQFLA,STAGEF                                             420
      COMMON FIRSTF                                                           430
      COMMON READF                                                            440
      COMMON EOFLAG                                                           450
C--------------                                                               460
C                                                                             470
      EQUIVALENCE(CNTRS(1),WIS)                                               480
      EQUIVALENCE(CNTRS(2),SENTEN)                                            490
      EQUIVALENCE(CNTRS(3),PARAGR)                                            500
      EQUIVALENCE(CNTRS(4),CHAPTE)                                            510
      EQUIVALENCE(CNTRS(5),VOLUME)                                            520
      EQUIVALENCE (BLANKS,WORDS(9,6))                                         530
C                                                                             540
C     PREFIX LENGTH.                                                          550
      DATA PREFL/0/                                                           560
      DATA PREFIX/,      ,/, IDIOM/0/                                         570
C                                                                             580
C     WE INCREMENT TEXT COUNTERS AFTER WRITING.                               590
```

```
1550    OUTCNT=OUTCNT+1                                                 600
        J1=WRDPTR                                                       610
1552    IF(J1 .GT. 18) GO TO 1551                                       620
        WORD(J1)=BLANKS                                                 630
        J1=J1+1                                                         640
        GO TO 1552                                                      650
1551    WORDL=WRDPTR-1                                                  660
        WRITE(OUTFIL)       WORDN,VOLUME,CHAPTE,PARAGR,SENTEN,WIS,      670
      * PAGEN,IDIOM,PREFL,WORDL,(PREFIX,J=1,8),WORD                     680
        IF( .NOT. PRINTF) GO TO 2030                                    690
        IF(WRDPTR .GT. 18) GO TO 11                                     700
        DO 10 J1=WRDPTR,18                                              710
10      WORD(J1)=BLANKS                                                 720
11      WRITE(SYSPRT,9021) WORD,VOLUME,                                 730
      *                            CHAPTE,PARAGR,SENTEN,WIS             740
9021    FORMAT(1H ,18A1,5X,1H(,I4,4(2H--,I4),1H))                       750
        LINES=LINES+1                                                   760
        IF(LINES .GE. LINEPG) CALL OFLO                                 770
2030    WIS=WIS+1                                                       780
        WORDN=WORDN+1                                                   790
        RETURN                                                          800
        END                                                             810
```

```
CPRINT      SUBR TO PRINT PROCESSING PARAMETERS.                              10
      SUBROUTINE PRINT (ALPHA)                                                20
      LOGICAL STAGED                                                          30
C                                                                             40
C-------------------------------------------------------------------------    50
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.               60
      INTEGER EOFLAG                                                          70
      LOGICAL READF                                                           80
      LOGICAL FIRSTF                                                          90
      LOGICAL PRINTF,SEQFLA,STAGEF                                           100
      INTEGER CONTIN                                                         110
      INTEGER WORDS(25,6)                                                    120
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       130
      INTEGER TYPEPR                                                         140
      INTEGER CARDCN,OUTCNT                                                  150
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     160
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            170
      INTEGER WRDPTR                                                         180
      COMMON /SYMBOL/ WORDS                                                  190
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    200
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         210
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        220
      COMMON PRINTF,SEQFLA,STAGEF                                            230
      COMMON FIRSTF                                                          240
      COMMON READF                                                           250
      COMMON EOFLAG                                                          260
C--------------                                                              270
C                                                                            280
C                                                                            290
      IF(FIRSTF) CALL ENDPAG                                                 300
      WRITE(SYSPRT,9000) ALPHA                                               310
      LINES=LINES+3                                                          320
9000  FORMAT(//,1H ,42HTHE FOLLOWING OPTIONS HAVE BEEN SPECIFIED ,           330
     * 12HOR ASSUMED- ,23HTYPE OF PROCESSING --- ,A4)                        340
      J1=9                                                                   350
      IF(PRINTF) J1=10                                                       360
      WRITE(SYSPRT,9001) (WORDS(J1,J2),J2=1,4)                               370
      LINES=LINES+1                                                          380
9001  FORMAT(/,1H 8HPRINTING,10X,5H --- ,4A1)                                390
      J1=9                                                                   400
      IF(SEQFLA) J1=10                                                       410
      WRITE(SYSPRT,9002) (WORDS(J1,J2),J2=1,4)                               420
      LINES=LINES+2                                                          430
9002  FORMAT(/,1H ,23HSEQUENCE CHECKING  --- ,4A1)                           440
      IF(STAGEF) GO TO 1380                                                  450
      WRITE(SYSPRT,9003) (WORDS(9,J2),J2=1,3)                                460
      LINES=LINES+2                                                          470
9003  FORMAT(/,1H ,23HSTAGE DIRECTIONS   --- ,3A1)                           480
      GO TO 999                                                              490
1380  WRITE(SYSPRT,9004) (WORDS(10,J2),J2=1,4)                               500
      LINES=LINES+2                                                          510
9004  FORMAT(/,1H ,23HSTAGE DIRECTIONS   --- ,4A1)                           520
999   WRITE(SYSPRT,9005) WORDS(7,6)                                          530
9005  FORMAT(/,, ,,,DELIMITER USED,,5X,,--- ,,A1)                            540
      CALL OFLO                                                              550
      RETURN                                                                 560
      END                                                                    570
```

```
CPROS       SUBROUTINE TO PROCESS ,PROS, TEXT.                                10
      SUBROUTINE PROS                                                         20
       INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                         30
       INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                         40
       INTEGER AMPERS                                                         50
       INTEGER BANG                                                           60
C                                                                             70
C                                                                             80
C-------------------------------------------------------------------------   90
C      THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.              100
       INTEGER EOFLAG                                                         110
       LOGICAL READF                                                          120
       LOGICAL FIRSTF                                                         130
       LOGICAL PRINTF,SEQFLA,STAGEF                                           140
       INTEGER CONTIN                                                         150
       INTEGER WORDS(25,6)                                                    160
       INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       170
       INTEGER TYPEPR                                                         180
       INTEGER CARDCN,OUTCNT                                                  190
       INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     200
       INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            210
       INTEGER WRDPTR                                                         220
       COMMON /SYMBOL/ WORDS                                                  230
       COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    240
       COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         250
       COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        260
       COMMON PRINTF,SEQFLA,STAGEF                                            270
       COMMON FIRSTF                                                          280
       COMMON READF                                                           290
       COMMON EOFLAG                                                          300
C--------------                                                               310
C                                                                             320
       EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                   330
       EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                  340
       EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                 350
       EQUIVALENCE (BLANKS,WORDS(9,6)  )                                      360
       EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                 370
       EQUIVALENCE (CNTRS(1),WIS)                                             380
       EQUIVALENCE (CNTRS(2),SENTEN)                                          390
       EQUIVALENCE (CNTRS(3),PARAGR)                                          400
       EQUIVALENCE (CNTRS(4),CHAPTE)                                          410
       EQUIVALENCE (CNTRS(5),VOLUME)                                          420
       EQUIVALENCE (BANG,WORDS(20,4))                                         430
C                                                                             440
C      INIT COMMON.                                                           450
       WORDN=1                                                                460
       VOLUME=0                                                               470
C      LIMIT OF DATA ON CARD.                                                 480
       LIMCOL=71                                                              490
       GO TO 1460                                                             500
C                                                                             510
C                                                                             520
C                                                                             530
C                                                                             540
C                                                                             550
C                                                                             560
C                                                                             570
C      TEST FOR RESETTING OF COUNTERS.                                        580
1140   IF(WORD(1) .NE. AMPERS) GO TO 1531                                     590
```

```
      DO 1151 J1=15,18                                                  600
      DO 1152 J2=1,6                                                    610
      IF(WORD(J2+1) .NE. WORDS(J1,J2)) GO TO 1151                       620
1152  CONTINUE                                                          630
C     GET NUMBER TO RESET COUNTER.                                      640
      WRDPTR=1                                                          650
      CALL GETWRD                                                       660
      CALL CONVRT(WORD,WRDPTR,NUM)                                      670
      I=J1-13                                                           680
      CNTRS(I)=NUM                                                      690
C     RESET ALL LOWER LEVEL COUNTERS.                                   700
1156  I=I-1                                                             710
      IF(I .LE. 1) GO TO 660                                            720
      CNTRS(I)=1                                                        730
      GO TO 1156                                                        740
1151  CONTINUE                                                          750
      GO TO 660                                                         760
C                                                                       770
C                                                                       780
1460  VOLUME=VOLUME+1                                                   790
1461  CHAPTE=1                                                          800
1462  PARAGR=1                                                          810
1463  SENTEN=1                                                          820
1464  WIS=1                                                             830
660   WRDPTR=1                                                          840
      CALL GETWRD                                                       850
C     CHECK FOR END OF INPUT DATA.                                      860
1530  IF(EOFLAG .EQ. 1) RETURN                                          870
1430  IF(WRDPTR .GT. 5) GO TO 1140                                      880
      IF(WORD(1) .NE. AMPERS) GO TO 1260                                890
      IF(WORD(2) .NE. AMPERS) GO TO 1260                                900
      IF(WORD(3) .NE. AMPERS) GO TO 1260                                910
      IF(WRDPTR .EQ. 4) GO TO 1350                                      920
      IF(WORD(4) .NE. AMPERS) GO TO 1260                                930
C     WORD = $$$$                                                       940
C     WE HAVE END OF VOLUME NOW.                                        950
      GO TO 1460                                                        960
C                                                                       970
C     WE HAVE END OF CHAPTER NOW.                                       980
C     WORD = $$$                                                        990
1350  CHAPTE=CHAPTE+1                                                  1000
      GO TO 1462                                                       1010
1260  IF(WRDPTR .NE. 3) GO TO 1291                                     1020
      IF(WORD(1) .NE. PERIOD) GO TO 1531                               1030
      IF(WORD(2) .NE. PERIOD) GO TO 1531                               1040
C     WORD = ..                                                        1050
C     WE HAVE END OF PARAGRAPH NOW.                                    1060
      WRDPTR=2                                                         1070
      CALL WRTWRD                                                      1080
1290  PARAGR=PARAGR+1                                                  1090
      GO TO 1463                                                       1100
1291  IF(WRDPTR .NE. 2) GO TO 1531                                     1110
      IF(WORD(1) .EQ. PERIOD) GO TO 1210                               1120
      IF(WORD(1) .EQ. QUESTM) GO TO 1210                               1130
      IF(WORD(1) .NE. BANG  ) GO TO 1531                               1140
C     WE HAVE END OF SENTENCE NOW.                                     1150
1210  WRDPTR=2                                                         1160
      CALL WRTWRD                                                      1170
1220  SENTEN=SENTEN+1                                                  1180
      GO TO 1464                                                       1190
```

```
1531    CALL WRTWRD                                                     1200
        GO TO 660                                                       1210
        END                                                             1220
```

```
CSPOK         SUBROUTINE TO PROCESS -SPOK- TEXT.                                10
      SUBROUTINE SPOK                                                           20
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                            30
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                            40
      INTEGER AMPERS                                                            50
      INTEGER BANG                                                              60
C                                                                               70
C                                                                               80
C-----------------------------------------------------------------------        90
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.                100
      INTEGER EOFLAG                                                           110
      LOGICAL READF                                                            120
      LOGICAL FIRSTF                                                           130
      LOGICAL PRINTF,SEQFLA,STAGEF                                             140
      INTEGER CONTIN                                                           150
      INTEGER WORDS(25,6)                                                      160
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                         170
      INTEGER TYPEPR                                                           180
      INTEGER CARDCN,OUTCNT                                                    190
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                       200
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                              210
      INTEGER WRDPTR                                                           220
      COMMON /SYMBOL/ WORDS                                                    230
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                      240
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN           250
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG          260
      COMMON PRINTF,SEQFLA,STAGEF                                              270
      COMMON FIRSTF                                                            280
      COMMON READF                                                             290
      COMMON EOFLAG                                                            300
C-------------                                                                 310
C                                                                              320
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                     330
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                    340
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                   350
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                        360
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                   370
      EQUIVALENCE (CNTRS(1),WIS)                                               380
      EQUIVALENCE (CNTRS(2),SENTEN)                                            390
      EQUIVALENCE (CNTRS(3),PARAGR)                                            400
      EQUIVALENCE (CNTRS(4),CHAPTE)                                            410
      EQUIVALENCE (CNTRS(5),VOLUME)                                            420
      EQUIVALENCE (BANG,WORDS(20,4))                                           430
C                                                                              440
C     INIT COMMON.                                                             450
      WORDN=1                                                                  460
      VOLUME=0                                                                 470
      CHAPTE=1                                                                 480
      PARAGR=1                                                                 490
      SENTEN=1                                                                 500
      WIS=1                                                                    510
C     LIMIT OF DATA ON CARD.                                                   520
      LIMCOL=71                                                                530
      GO TO 1460                                                               540
C                                                                              550
C                                                                              560
1140  DO 1141 J1=1,6                                                           570
      IF(WORD(J1+1) .NE. WORDS(22,J1)) GO TO 1150                              580
1141  CONTINUE                                                                 590
```

```
C                                                                       600
C       NEW SERIES.                                                     610
        WRDPTR=1                                                        620
        CALL GETWRD                                                     630
        IF(EOFLAG .EQ. 1) RETURN                                        640
        CALL CONVRT(WORD,WRDPTR,NUM)                                    650
        VOLUME=NUM                                                      660
        GO TO 1461                                                      670
C                                                                       680
1150    DO 1151 J1=1,6                                                  690
        IF(WORD(J1+1) .NE. WORDS(23,J1)) GO TO 1160                     700
1151    CONTINUE                                                        710
C                                                                       720
C       NEW SESSION.                                                    730
        WRDPTR=1                                                        740
        CALL GETWRD                                                     750
        CALL CONVRT(WORD,WRDPTR,NUM)                                    760
        CHAPTE=NUM                                                      770
        GO TO 1462                                                      780
C                                                                       790
1160    DO 1161 J1=1,6                                                  800
        IF(WORD(J1+1) .NE. WORDS(24,J1)) GO TO 1531                     810
1161    CONTINUE                                                        820
C                                                                       830
C       NEW SPEAKER.                                                    840
        WRDPTR=1                                                        850
        CALL GETWRD                                                     860
        CALL CONVRT(WORD,WRDPTR,NUM)                                    870
        PARAGR=NUM                                                      880
        GO TO 1463                                                      890
C                                                                       900
1460   VOLUME=VOLUME+1                                                  910
1461    CHAPTE=1                                                        920
1462    PARAGR=1                                                        930
1463    SENTEN=1                                                        940
1464    WIS=1                                                           950
660     WRDPTR=1                                                        960
        CALL GETWRD                                                     970
        IF(EOFLAG .EQ. 1) RETURN                                        980
1430    IF(WORD(1) .NE. AMPERS) GO TO 1260                              990
        IF(WORD(2) .NE. AMPERS) GO TO 1140                             1000
        IF(WORD(3) .NE. AMPERS) GO TO 1260                             1010
        IF(WRDPTR .EQ. 4) GO TO 1370                                   1020
        IF(WORD(4) .NE. AMPERS) GO TO 1250                             1030
C       WORD = $$$$                                                    1040
C       WE HAVE END OF VOLUME NOW.                                     1050
        GO TO 1450                                                     1060
C                                                                      1070
C       WE HAVE END OF CHAPTER NOW.                                    1080
C       WORD = $$$                                                     1090
1370    CHAPTE=CHAPTE+1                                                1100
        GO TO 1462                                                     1110
1250    IF(WRDPTR .NE. 3) GO TO 1291                                   1120
        IF(WORD(1) .NE. PERIOD) GO TO 1531                             1130
        IF(WORD(2) .NE. PERIOD) GO TO 1531                             1140
C       WORD = ..                                                      1150
C       WE HAVE END OF PARAGRAPH NOW.                                  1160
        WRDPTR=2                                                       1170
        CALL WRTWRD                                                    1180
        PARAGR=PARAGR+1                                                1190
```

```
      GO TO 1463                                                        1200
C                                                                       1210
1291  IF(WRDPTR .NE. 2) GO TO 1531                                      1220
      IF(WORD(1) .EQ. PERIOD) GO TO 1210                                1230
      IF(WORD(1) .EQ. QUESTN) GO TO 1210                                1240
      IF(WORD(1) .NE. BANG) GO TO 1531                                  1250
C     WE HAVE END OF SENTENCE NOW.                                      1260
1210  WRDPTR=2                                                          1270
      CALL WRTWRD                                                       1280
1220  SENTEN=SENTEN+1                                                   1290
      GO TO 1464                                                        1300
1531  CALL WRTWRD                                                       1310
      GO TO 660                                                         1320
      END                                                               1330
```

```
CPLAY       SUBROUTINE TO PROCESS -PLAY- TEXT.                              10
      SUBROUTINE PLAY                                                       20
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                        30
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                        40
      INTEGER AMPERS                                                        50
      INTEGER NUMBRS(10)                                                    60
      INTEGER STAR                                                          70
      INTEGER BANG                                                          80
C                                                                           90
C                                                                           100
C-------------------------------------------------------------------------  110
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.             120
      INTEGER EOFLAG                                                        130
      LOGICAL READF                                                         140
      LOGICAL FIRSTF                                                        150
      LOGICAL PRINTF,SEQFLA,STAGEF                                          160
      INTEGER CONTIN                                                        170
      INTEGER WORDS(25,6)                                                   180
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                      190
      INTEGER TYPEPR                                                        200
      INTEGER CARDCN,OUTCNT                                                 210
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                    220
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                           230
      INTEGER WRDPTR                                                        240
      COMMON /SYMBOL/ WORDS                                                 250
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                   260
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN        270
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG       280
      COMMON PRINTF,SEQFLA,STAGEF                                           290
      COMMON FIRSTF                                                         300
      COMMON READF                                                          310
      COMMON EOFLAG                                                         320
C-------------                                                              330
C                                                                           340
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                  350
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                 360
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                370
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                     380
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                390
      EQUIVALENCE (CNTRS(1),WIS)                                            400
      EQUIVALENCE (CNTRS(2),SENTEN)                                         410
      EQUIVALENCE (CNTRS(3),PARAGR)                                         420
      EQUIVALENCE (CNTRS(4),CHAPTE)                                         430
      EQUIVALENCE (CNTRS(5),VOLUME)                                         440
      EQUIVALENCE (STAR,WORDS(11,6))                                        450
      EQUIVALENCE (BANG,WORDS(20,4))                                        460
C                                                                           470
      DATA NUMBRS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/                  480
C     INIT COMMON.                                                          490
      WORDN=1                                                               500
      VOLUME=1                                                              510
      CHAPTE=1                                                              520
      PARAGR=1                                                              530
      SENTEN=1                                                              540
C     IF WE ARE PROCESSING VERSE PLAY, THE FIRST READ WILL INCR             550
C     SENTEN COUNTER.                                                       560
      IF(TYPEPR .EQ. 5) SENTEN=0                                            570
C     LIMIT OF DATA ON CARD.                                                580
      LIMCOL=71                                                             590
```

```
      GO TO 1464                                                        600
C                                                                       610
1600  VOLUME=VOLUME+1                                                   620
1461  CHAPTE=1                                                          630
1462  PARAGR=1                                                          640
1463  SENTEN=1                                                          650
1464  WIS=1                                                             660
C                                                                       670
660   WRDPTR=1                                                          680
662   CALL GETWRD                                                       690
      IF(EOFLAG .EQ. 1) RETURN                                          700
C     IS THIS A STAGE DIRECTION ---                                     710
      IF(WORD(1) .NE. STAR) GO TO 1429                                  720
C     YES, SHOULD IT BE IN OUTPUT DATASET ---                           730
      IF( .NOT. STAGEF) GO TO 662                                       740
      GO TO 1531                                                        750
C                                                                       760
C     ARE WE PROCESSING VERSE PLAY ---                                  770
1429  IF(TYPEPR .NE. 5) GO TO 1430                                      780
C     IS THIS A NEW LINE ---                                            790
      IF( .NOT. READF) GO TO 1430                                       800
      SENTEN=SENTEN+1                                                   810
      WIS=1                                                             820
1430  IF(WORD(1) .NE. AMPERS) GO TO 1260                                830
      IF(WORD(2) .NE. AMPERS) GO TO 1700                                840
      IF(WORD(3) .NE. AMPERS) GO TO 1531                                850
      IF(WORD(4) .EQ. AMPERS) GO TO 1600                                860
C     WORD=$$$                                                          870
C     WE HAVE END OF SCENE NOW.                                         880
1350  WRDPTR=1                                                          890
1370  CHAPTE=CHAPTE+1                                                   900
      GO TO 1462                                                        910
1260  IF(WORD(1) .NE. PERIOD) GO TO 1291                                920
      IF(WORD(2) .NE. PERIOD) GO TO 1291                                930
C     WORD = ..                                                         940
C     WE HAVE END OF PARAGRAPH NOW.                                     950
      WRDPTR=2                                                          960
      CALL WRTWRD                                                       970
1290  PARAGR=PARAGR+1                                                   980
      GO TO 1463                                                        990
C                                                                      1000
1700  DO 1701 J1=1,3                                                   1010
      IF(WORD(J1+1) .NE. WORDS(20,J1)) GO TO 1710                      1020
1701  CONTINUE                                                         1030
      WRDPTR=1                                                         1040
      CALL GETWRD                                                      1050
      IF(EOFLAG .EQ. 1) RETURN                                         1060
      CALL CONVRT(WORD,WRDPTR,NUM)                                     1070
C     RESET ACT.                                                       1080
      VOLUME=NUM                                                       1090
      GO TO 1461                                                       1100
1710  DO 1711 J1=1,5                                                   1110
      IF(WORD(J1+1) .NE. WORDS(21,J1)) GO TO 1531                      1120
1711  CONTINUE                                                         1130
      WRDPTR=1                                                         1140
      CALL GETWRD                                                      1150
      CALL CONVRT(WORD,WRDPTR,NUM)                                     1160
C     RESET SCENE.                                                     1170
      CHAPTE=NUM                                                       1180
      GO TO 1462                                                       1190
```

```
C                                                                       1200
1291    IF(WRDPTR .NE. 2) GO TO 1531                                    1210
        IF(WORD(1) .EQ. PERIOD) GO TO 1210                              1220
        IF(WORD(1) .EQ. QUESTM) GO TO 1210                              1230
        IF(WORD(1) .NE.  BANG ) GO TO 1531                              1240
C       WE HAVE END OF SENTENCE NOW.                                    1250
1210    WRDPTR=2                                                        1260
        CALL WRTWRD                                                     1270
1220    SENTEN=SENTEN+1                                                 1280
        GO TO 1464                                                      1290
1531    CALL WRTWRD                                                     1300
        GO TO 660                                                       1310
        END                                                             1320
```

```
CGETWRD     SUBROUTINE TO GET TEXT WORD FROM INPUT DATA.                    10
      SUBROUTINE GETWRD                                                     20
C     THIS IS GETWORD ROUTINE.                                              30
C     CANNOT LOCAL THIS SUBR.                                               40
C                                                                           50
C     P1 POINTS TO THE FIRST CHAR OF THE WORD IN THE TEXT STRING.           60
C     P2 POINTS TO THE LAST CHAR OF THE WORD IN THE TEXT STRING.            70
C     WRDPTR POINTS TO THE (LAST CHAR + 1) IN THE ,WORD, ARRAY.             80
      INTEGER P1,P2                                                         90
      INTEGER BLANKS,DASH                                                  100
      INTEGER ATSIGN                                                       110
C-------------------------------------------------------------------       120
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.            130
      INTEGER EOFLAG                                                       140
      LOGICAL READF                                                        150
      LOGICAL FIRSTF                                                       160
      LOGICAL PRINTF,SEQFLA,STAGEF                                         170
      INTEGER CONTIN                                                       180
      INTEGER WORDS(25,6)                                                  190
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                     200
      INTEGER TYPEPR                                                       210
      INTEGER CARDCN,OUTCNT                                                220
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                   230
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                          240
      INTEGER WRDPTR                                                       250
      COMMON /SYMBOL/ WORDS                                                260
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                  270
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN       280
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG      290
      COMMON PRINTF,SEQFLA,STAGEF                                          300
      COMMON FIRSTF                                                        310
      COMMON READF                                                         320
      COMMON EOFLAG                                                        330
C-------------                                                             340
C                                                                          350
```

```
CMAIN       MAINLINE                                                          10
C                                                                             20
C       SUBROUTINES THAT ARE SOFTWARE AND MACHINE DEPENDENT --                30
C       CLOCK1                                                                40
C       FLGEOF                                                                50
C                                                                             60
C                                                                             70
C       CORRESPONDENCE OF COUNTERS FOR VARIOUS PROCESSING MODES -             80
C        -PROS-     -MILT-     -PLAY-     -VPLA-     -POET-     -SPOK-       90
C          VOLUME     VOLUME     ACT        ACT        1          SERIES     100
C          CHAPTER    BOOK       SCENE      SCENE      1          SESSION    110
C          PARAGRAPH  PARAGRAPH  PARAGRAPH  PARAGRAPH  STANZA     SPEAKER    120
C          SENTENCE   LINE       SENTENCE   LINE       LINE       SENTENCE   130
C          WORD       WORD       WORD       WORD       WORD       WORD       140
C                                                                            150
C                                                                            160
      INTEGER COMMA,BLANKS                                                   170
      INTEGER PARAMS(80)                                                     180
      INTEGER AMPERS                                                         190
C-------------------------------------------------------------------------   200
C       THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.            210
      INTEGER EOFLAG                                                         220
      LOGICAL READF                                                          230
      LOGICAL FIRSTF                                                         240
      LOGICAL PRINTF,SEQFLA,STAGEF                                           250
      INTEGER CONTIN                                                         260
      INTEGER WORDS(25,6)                                                    270
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       280
      INTEGER TYPEPR                                                         290
      INTEGER CARDCN,OUTCNT                                                  300
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     310
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            320
      INTEGER WRDPTR                                                         330
      COMMON /SYMBOL/ WORDS                                                  340
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    350
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         360
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        370
      COMMON PRINTF,SEQFLA,STAGEF                                            380
      COMMON FIRSTF                                                          390
      COMMON READF                                                           400
      COMMON EOFLAG                                                          410
C-------------------------------------------------------------------------   420
C                                                                            430
      EQUIVALENCE (COMMA,WORDS(9,3))                                         440
      EQUIVALENCE (BLANKS,WORDS(9,6))                                        450
      EQUIVALENCE (AMPERS,WORDS(7,6))                                        460
C                                                                            470
      DATA MASK77/O777777777777/                                             480
C                                                                            490
C                                                                            500
C       INITIALIZE BLANK COMMON VARIABLES.                                   510
      CALL INITCM                                                            520
C                                                                            530
C       INITIALIZE EOFLAG.                                                   540
C       EOFLAG - 0 OK, 1 EOF CONDITION READ.                                 550
      CALL FLGEOF(INFILE,EOFLAG)                                             560
C                                                                            570
C       GET TODAYS DATE.                                                     580
      CALL CLOCK1(IDAY,IMON,IYR)                                             590
```

```
      READ(SYSIN,9002) PARAMS                                          600
9002  FORMAT(80A1)                                                     610
      READ(SYSIN,9000) TITLE                                           620
9000  FORMAT(20A4)                                                     630
C     INSERT COMMA AT END OF PARM STRING.                              640
460   IPTR=81                                                          650
461   IPTR=IPTR-1                                                      660
      IF(IPTR) 1420,1420,462                                           670
462   IF(PARAMS(IPTR) .EQ. BLANKS) GO TO 461                           680
      PARAMS(IPTR+1)=COMMA                                             690
C     SCAN DOWN TO FIRST NONBLANK COLUMN.                              700
      DO 463 J1=1,80                                                   710
      IF(PARAMS(J1) .NE. BLANKS) GO TO 464                             720
463   CONTINUE                                                         730
      GO TO 1420                                                       740
C     IPTR POINTS TO LAST COLUMN LOOKED AT.                            750
464   IPTR=J1-1                                                        760
C     IS FIRST PARM ,PROC=, ---                                        770
469   DO 470 J1=1,5                                                    780
      L=J1+IPTR                                                        790
      IF(PARAMS(L) .NE. WORDS(1,J1)) GO TO 590                         800
470   CONTINUE                                                         810
C     FIND OUT KEYWORD - (PROS,POET,PLAY,VPLA,MILT,SPOK).              820
C              TYPEPR --   (2,3,4,5,6,7)                               830
      DO 540 ITYPE=2,7                                                 840
      DO 541 J1=1,5                                                    850
      L=J1+IPTR                                                        860
      IF(PARAMS(L+5) .NE. WORDS( ITYPE,J1)) GO TO 540                  870
541   CONTINUE                                                         880
C     REMEMBER TYPE OF PROCESSING.                                     890
      TYPEPR=ITYPE                                                     900
      GO TO 560                                                        910
540   CONTINUE                                                         920
      GO TO 990                                                        930
560   J=10                                                             940
      GO TO 1050                                                       950
C                                                                      960
C     IS NEXT PARM ,PRINT=, ---                                        970
590   DO 591 J1=1,6                                                    980
      L=J1+IPTR                                                        990
      IF(PARAMS(L ) .NE. WORDS(8,J1)) GO TO 720                       1000
591   CONTINUE                                                        1010
C     IS PRINT KEYWORD ,NO,, ---                                      1020
      DO 600 J1=1,3                                                   1030
      L=J1+IPTR                                                       1040
      IF(PARAMS(L +6) .NE. WORDS(9,J1)) GO TO 650                     1050
600   CONTINUE                                                        1060
      PRINTF=.FALSE.                                                  1070
      J=9                                                             1080
      GO TO 1050                                                      1090
C     IS PRINT KEYWORD ,YES,, ---                                     1100
650   DO 651 J1=1,4                                                   1110
      L=J1+IPTR                                                       1120
      IF(PARAMS(L +6) .NE. WORDS(10,J1)) GO TO 990                    1130
651   CONTINUE                                                        1140
      PRINTF=.TRUE.                                                   1150
      J=10                                                            1160
      GO TO 1050                                                      1170
C                                                                     1180
C     IS NEXT PARM ,SEQ=, ---                                         1190
```

```
720     DO 721 J1=1,4                                                   1200
        L=J1+IPTR                                                       1210
        IF(PARAMS(L ) .NE. WORDS(11,J1)) GO TO 850                      1220
721     CONTINUE                                                        1230
C       IS SEQ KEYWORD ,NO,, ---                                        1240
        DO 730 J1=1,3                                                   1250
        L=J1+IPTR                                                       1260
        IF(PARAMS(L +4) .NE. WORDS(9,J1)) GO TO 780                     1270
730     CONTINUE                                                        1280
        SEQFLA=.FALSE.                                                  1290
        J=7                                                             1300
        GO TO 1050                                                      1310
C       IS SEQ KEYWORD ,YES,, ---                                       1320
780     DO 781 J1=1,4                                                   1330
        L=J1+IPTR                                                       1340
        IF(PARAMS(L +4) .NE. WORDS(10,J1)) GO TO 990                    1350
781     CONTINUE                                                        1360
        SEQFLA=.TRUE.                                                   1370
        J=8                                                             1380
        GO TO 1050                                                      1390
C                                                                       1400
C       IS NEXT PARM ,STAGE=, ---                                       1410
850     DO 851 J1=1,6                                                   1420
        L=J1+IPTR                                                       1430
        IF(PARAMS(L ) .NE. WORDS(12,J1)) GO TO 870                      1440
851     CONTINUE                                                        1450
C       IS STAGE KEYWORD ,NO,, ---                                      1460
        DO 860 J1=1,3                                                   1470
        L=J1+IPTR                                                       1480
        IF(PARAMS(L +6) .NE. WORDS(9,J1)) GO TO 910                     1490
860     CONTINUE                                                        1500
        STAGEF=.FALSE.                                                  1510
        J=9                                                             1520
        GO TO 1050                                                      1530
C       IS STAGE KEYWORD ,YES,, ---                                     1540
910     DO 911 J1=1,4                                                   1550
        L=J1+IPTR                                                       1560
        IF(PARAMS(L +6) .NE. WORDS(10,J1)) GO TO 990                    1570
911     CONTINUE                                                        1580
        STAGEF=.TRUE.                                                   1590
        J=10                                                            1600
        GO TO 1050                                                      1610
C                                                                       1620
C       IS NEXT PARM ,DELIM=, ---                                       1630
870     DO 871 J1=1,6                                                   1640
        L=J1+IPTR                                                       1650
        IF(PARAMS(L) .NE. WORDS(25,J1)) GO TO 990                       1660
871     CONTINUE                                                        1670
C       SUBSTITUTE DELIMITER.                                           1680
        AMPERS=PARAMS(L+1)                                              1690
        J=8                                                             1700
        GO TO 1050                                                      1710
C                                                                       1720
C       ERROR ROUTINE.                                                  1730
990     LIM1=IPTR+1                                                     1740
C       SEARCH FOR COMMA AT END OF PARM IN ERROR.                       1750
        DO 991 J=LIM1,80                                                1760
        IF(PARAMS(J) .EQ. COMMA) GO TO 992                              1770
991     CONTINUE                                                        1780
        GO TO 1420                                                      1790
```

```
992     LIM2=J                                                                  1800
C                                                                               1810
        IF(FIRSTF) CALL ENDPAG                                                  1820
        WRITE(SYSPRT,9001) (PARAMS(J1),J1=LIM1,LIM2)                            1830
9001    FORMAT(//,1H ,8HKEYWORD ,     1X,19HINVALID OR IN ERROR,                1840
     * 3H - ,80A1)                                                             1850
        IPTR=J                                                                  1860
        GO TO 1051                                                              1870
C                                                                               1880
C       BUMP PAST KEYWORD WE JUST FOUND.                                        1890
1050    IPTR=IPTR+J                                                             1900
C       ARE THERE MORE PARMS ---                                                1910
1051    IF(PARAMS(IPTR+1) .NE. BLANKS) GO TO 469                                1920
1070    GO TO (1420,1080,1110,1230,1140,1170,1200),TYPEPR                       1930
C                                                                               1940
1080    CALL PRINT(4HPROS)                                                      1950
        CALL PROS                                                               1960
        GO TO 1480                                                              1970
C                                                                               1980
1110    CALL PRINT(4HPOET)                                                      1990
        CALL POET                                                               2000
        GO TO 1480                                                              2010
C                                                                               2020
1140    CALL PRINT(4HVPLA)                                                      2030
        CALL PLAY                                                               2040
        GO TO 1480                                                              2050
C                                                                               2060
1170    CALL PRINT(4HMILT)                                                      2070
        CALL MILT                                                               2080
        GO TO 1480                                                              2090
C                                                                               2100
1200    CALL PRINT(4HSPOK)                                                      2110
        CALL SPOK                                                               2120
        GO TO 1480                                                              2130
C                                                                               2140
1230    CALL PRINT(4HPLAY)                                                      2150
        CALL PLAY                                                               2160
        GO TO 1480                                                              2170
C                                                                               2180
C       ERROR ROUTINE.                                                          2190
1420    IF(FIRSTF) CALL ENDPAG                                                  2200
        WRITE(SYSPRT,9020)                                                      2210
9020    FORMAT(//,1H ,46HTYPE OF PROCESSING WAS NOT SPECIFIED OR WAS IN,        2220
     * 7H ERROR.,/,1H ,32HJOB TERMINATED AT THIS POINT ***,//)                  2230
C                                                                               2240
C       PUT OUT EOF INDICATION ON ,OUTFIL,.                                     2250
1480    WORDN=0                                                                 2260
C       THIS CHARACTER WILL ALWAYS SORT TO THE END OF FILE.                     2270
        DO 998 J1=1,18                                                          2280
998     WORD(J1)=MASK77                                                         2290
        WRDPTR=19                                                               2300
        CALL WRTWRD                                                             2310
        END FILE OUTFIL                                                         2320
        REWIND OUTFIL                                                           2330
        REWIND INFILE                                                           2340
C                                                                               2350
        CALL ENDPAG                                                             2360
        WRITE(SYSPRT,9021) CARDCN,OUTCNT                                        2370
9021    FORMAT(//,1H ,I6,5X,11HCARDS INPUT,//,1H ,I6,5X,                        2380
     * 14HRECORDS OUTPUT,//,30X,22H***** END OF JOB *****)                      2390
```

```
C                                                                       2400
      CALL EXIT                                                         2410
      END                                                               2420
```

```
CWRTWRD     SUBROUTINE TO WRITE INDEXED WORD.                                  10
      SUBROUTINE WRTWRD                                                        20
C                                                                              30
C                                                                              40
C     OUTPUT RECORD FORMAT -                                                   50
C       WORD       LENGTH    FORMAT    FIELD                                  60
C       1          1         I         LINEAR WORD NUMBER IN TEXT             70
C       2          1         I         VOLUME                                 80
C       3          1         I         CHAPTER                                90
C       4          1         I         PARAGRAPH                             100
C       5          1         I         SENTENCE                              110
C       6          1         I         WORD-IN-SENTENCE                      120
C       7          1         I         PAGE NUMBER                           130
C       8          1         I         IDIOM FLAG                            140
C       9          1         I         PREFIX LENGTH IN CHARACTERS           150
C       10         1         I         WORD LENGTH IN CHARACTERS             160
C       11-18      8         A1        PREFIX CHARACTERS                     170
C       19-36      18        A1        TEXT WORD CHARACTERS                  180
C                                                                             190
C                                                                             200
      INTEGER BLANKS                                                          210
      INTEGER PREFL                                                           220
      INTEGER WIS,SENTEN,PARAGR,CHAPTE,VOLUME                                 230
C-----------------------------------------------------------------------      240
      INTEGER EOFLAG                                                          250
      LOGICAL READF                                                           260
      LOGICAL FIRSTF                                                          270
      LOGICAL PRINTF,SEQFLA,STAGEF                                            280
      INTEGER CONTIN                                                          290
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.               300
      INTEGER WORDS(25,6)                                                     310
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                        320
      INTEGER TYPEPR                                                          330
      INTEGER CARDCN,OUTCNT                                                   340
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                      350
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                             360
      INTEGER WRDPTR                                                          370
      COMMON /SYMBOL/ WORDS                                                   380
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                     390
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN          400
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG         410
      COMMON PRINTF,SEQFLA,STAGEF                                             420
      COMMON FIRSTF                                                           430
      COMMON READF                                                            440
      COMMON EOFLAG                                                           450
C--------------                                                               460
C                                                                             470
      EQUIVALENCE(CNTRS(1),WIS)                                               480
      EQUIVALENCE(CNTRS(2),SENTEN)                                            490
      EQUIVALENCE(CNTRS(3),PARAGR)                                            500
      EQUIVALENCE(CNTRS(4),CHAPTE)                                            510
      EQUIVALENCE(CNTRS(5),VOLUME)                                            520
      EQUIVALENCE (BLANKS,WORDS(9,6))                                         530
C                                                                             540
C     PREFIX LENGTH.                                                          550
      DATA PREFL/0/                                                           560
      DATA PREFIX/,      ,/, IDIOM/0/                                         570
C                                                                             580
C     WE INCREMENT TEXT COUNTERS AFTER WRITING.                               590
```

```
1550    OUTCNT=OUTCNT+1                                                 600
        J1=WRDPTR                                                       610
1552    IF(J1 .GT. 18) GO TO 1551                                       620
        WORD(J1)=BLANKS                                                 630
        J1=J1+1                                                         640
        GO TO 1552                                                      650
1551    WORDL=WRDPTR-1                                                  660
        WRITE(OUTFIL)       WORDN,VOLUME,CHAPTE,PARAGR,SENTEN,WIS,      670
     *  PAGEN,IDIOM,PREFL,WORDL,(PREFIX,J=1,8),WORD                     680
        IF( .NOT. PRINTF) GO TO 2030                                    690
        IF(WRDPTR .GT. 18) GO TO 11                                     700
        DO 10 J1=WRDPTR,18                                              710
10      WORD(J1)=BLANKS                                                 720
11      WRITE(SYSPRT,9021) WORD,VOLUME,                                 730
     *                             CHAPTE,PARAGR,SENTEN,WIS           740
9021    FORMAT(1H ,18A1,5X,1H(,I4,4(2H--,I4),1H))                       750
        LINES=LINES+1                                                   760
        IF(LINES .GE. LINEPG) CALL OFLO                                 770
2030    WIS=WIS+1                                                       780
        WORDN=WORDN+1                                                   790
        RETURN                                                          800
        END                                                             810
```

```
CPRINT      SUBR TO PRINT PROCESSING PARAMETERS.                              10
      SUBROUTINE PRINT (ALPHA)                                                20
      LOGICAL STAGED                                                          30
C                                                                             40
C-------------------------------------------------------------------------    50
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.               60
      INTEGER EOFLAG                                                          70
      LOGICAL READF                                                           80
      LOGICAL FIRSTF                                                          90
      LOGICAL PRINTF,SEQFLA,STAGEF                                           100
      INTEGER CONTIN                                                         110
      INTEGER WORDS(25,6)                                                    120
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       130
      INTEGER TYPEPR                                                         140
      INTEGER CARDCN,OUTCNT                                                  150
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     160
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            170
      INTEGER WRDPTR                                                         180
      COMMON /SYMBOL/ WORDS                                                  190
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    200
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         210
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        220
      COMMON PRINTF,SEQFLA,STAGEF                                            230
      COMMON FIRSTF                                                          240
      COMMON READF                                                           250
      COMMON EOFLAG                                                          260
C--------------                                                              270
C                                                                            280
C                                                                            290
      IF(FIRSTF) CALL ENDPAG                                                 300
      WRITE(SYSPRT,9000) ALPHA                                               310
      LINES=LINES+3                                                          320
9000  FORMAT(//,1H ,42HTHE FOLLOWING OPTIONS HAVE BEEN SPECIFIED ,           330
     * 12HOR ASSUMED- ,23HTYPE OF PROCESSING --- ,A4)                        340
      J1=9                                                                   350
      IF(PRINTF) J1=10                                                       360
      WRITE(SYSPRT,9001) (WORDS(J1,J2),J2=1,4)                               370
      LINES=LINES+1                                                          380
9001  FORMAT(/,1H 8HPRINTING,10X,5H --- ,4A1)                                390
      J1=9                                                                   400
      IF(SEQFLA) J1=10                                                       410
      WRITE(SYSPRT,9002) (WORDS(J1,J2),J2=1,4)                               420
      LINES=LINES+2                                                          430
9002  FORMAT(/,1H ,23HSEQUENCE CHECKING  --- ,4A1)                           440
      IF(STAGEF) GO TO 1380                                                  450
      WRITE(SYSPRT,9003) (WORDS(9,J2),J2=1,3)                                460
      LINES=LINES+2                                                          470
9003  FORMAT(/,1H ,23HSTAGE DIRECTIONS   --- ,3A1)                           480
      GO TO 999                                                              490
1380  WRITE(SYSPRT,9004) (WORDS(10,J2),J2=1,4)                               500
      LINES=LINES+2                                                          510
9004  FORMAT(/,1H ,23HSTAGE DIRECTIONS   --- ,4A1)                           520
999   WRITE(SYSPRT,9005) WORDS(7,6)                                          530
9005  FORMAT(/,, ,,,DELIMITER USED,,5X,,--- ,,A1)                            540
      CALL OFLO                                                              550
      RETURN                                                                 560
      END                                                                    570
```

```
CPROS       SUBROUTINE TO PROCESS ,PROS, TEXT.                                  10
      SUBROUTINE PROS                                                           20
       INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                           30
       INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                           40
       INTEGER AMPERS                                                           50
       INTEGER BANG                                                             60
C                                                                               70
C                                                                               80
C-----------------------------------------------------------------------        90
C      THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.                100
       INTEGER EOFLAG                                                           110
       LOGICAL READF                                                            120
       LOGICAL FIRSTF                                                           130
       LOGICAL PRINTF,SEQFLA,STAGEF                                             140
       INTEGER CONTIN                                                           150
       INTEGER WORDS(25,6)                                                      160
       INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                         170
       INTEGER TYPEPR                                                           180
       INTEGER CARDCN,OUTCNT                                                    190
       INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                       200
       INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                              210
       INTEGER WRDPTR                                                           220
       COMMON /SYMBOL/ WORDS                                                    230
       COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                      240
       COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN           250
       COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG          260
       COMMON PRINTF,SEQFLA,STAGEF                                              270
       COMMON FIRSTF                                                            280
       COMMON READF                                                             290
       COMMON EOFLAG                                                            300
C--------------                                                                 310
C                                                                               320
       EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                     330
       EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                    340
       EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                   350
       EQUIVALENCE (BLANKS,WORDS(9,6)  )                                        360
       EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                   370
       EQUIVALENCE (CNTRS(1),WIS)                                               380
       EQUIVALENCE (CNTRS(2),SENTEN)                                            390
       EQUIVALENCE (CNTRS(3),PARAGR)                                            400
       EQUIVALENCE (CNTRS(4),CHAPTE)                                            410
       EQUIVALENCE (CNTRS(5),VOLUME)                                            420
       EQUIVALENCE (BANG,WORDS(20,4))                                           430
C                                                                               440
C      INIT COMMON.                                                             450
       WORDN=1                                                                  460
       VOLUME=0                                                                 470
C      LIMIT OF DATA ON CARD.                                                   480
       LIMCOL=71                                                                490
       GO TO 1460                                                               500
C                                                                               510
C                                                                               520
C                                                                               530
C                                                                               540
C                                                                               550
C                                                                               560
C                                                                               570
C      TEST FOR RESETTING OF COUNTERS.                                          580
1140   IF(WORD(1) .NE. AMPERS) GO TO 1531                                       590
```

```
      DO 1151 J1=15,18                                                  600
      DO 1152 J2=1,6                                                    610
      IF(WORD(J2+1) .NE. WORDS(J1,J2)) GO TO 1151                       620
1152  CONTINUE                                                          630
C     GET NUMBER TO RESET COUNTER.                                      640
      WRDPTR=1                                                          650
      CALL GETWRD                                                       660
      CALL CONVRT(WORD,WRDPTR,NUM)                                      670
      I=J1-13                                                           680
      CNTRS(I)=NUM                                                      690
C     RESET ALL LOWER LEVEL COUNTERS.                                   700
1156  I=I-1                                                             710
      IF(I .LE. 1) GO TO 660                                            720
      CNTRS(I)=1                                                        730
      GO TO 1156                                                        740
1151  CONTINUE                                                          750
      GO TO 660                                                         760
C                                                                       770
C                                                                       780
1460  VOLUME=VOLUME+1                                                   790
1461   CHAPTE=1                                                         800
1462   PARAGR=1                                                         810
1463   SENTEN=1                                                         820
1464   WIS=1                                                            830
660    WRDPTR=1                                                         840
       CALL GETWRD                                                      850
C      CHECK FOR END OF INPUT DATA.                                     860
1530   IF(EOFLAG .EQ. 1) RETURN                                         870
1430   IF(WRDPTR .GT. 5) GO TO 1140                                     880
       IF(WORD(1) .NE. AMPERS) GO TO 1260                               890
       IF(WORD(2) .NE. AMPERS) GO TO 1260                               900
       IF(WORD(3) .NE. AMPERS) GO TO 1260                               910
       IF(WRDPTR .EQ. 4) GO TO 1350                                     920
       IF(WORD(4) .NE. AMPERS) GO TO 1260                               930
C      WORD = $$$$                                                      940
C      WE HAVE END OF VOLUME NOW.                                       950
       GO TO 1460                                                       960
C                                                                       970
C      WE HAVE END OF CHAPTER NOW.                                      980
C      WORD = $$$                                                       990
1350   CHAPTE=CHAPTE+1                                                 1000
       GO TO 1462                                                      1010
1260   IF(WRDPTR .NE. 3) GO TO 1291                                    1020
       IF(WORD(1) .NE. PERIOD) GO TO 1531                              1030
       IF(WORD(2) .NE. PERIOD) GO TO 1531                              1040
C      WORD = ..                                                       1050
C      WE HAVE END OF PARAGRAPH NOW.                                   1060
       WRDPTR=2                                                        1070
       CALL WRTWRD                                                     1080
1290   PARAGR=PARAGR+1                                                 1090
       GO TO 1463                                                      1100
1291   IF(WRDPTR .NE. 2) GO TO 1531                                    1110
       IF(WORD(1) .EQ. PERIOD) GO TO 1210                              1120
       IF(WORD(1) .EQ. QUESTM) GO TO 1210                              1130
       IF(WORD(1) .NE. BANG  ) GO TO 1531                              1140
C      WE HAVE END OF SENTENCE NOW.                                    1150
1210   WRDPTR=2                                                        1160
       CALL WRTWRD                                                     1170
1220   SENTEN=SENTEN+1                                                 1180
       GO TO 1464                                                      1190
```

```
1531    CALL WRTWRD                                                    1200
        GO TO 660                                                      1210
        END                                                            1220
```

```
CSPOK         SUBROUTINE TO PROCESS -SPOK- TEXT.                                10
      SUBROUTINE SPOK                                                           20
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                            30
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                            40
      INTEGER AMPERS                                                            50
      INTEGER BANG                                                              60
C                                                                               70
C                                                                               80
C-----------------------------------------------------------------              90
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.                100
      INTEGER EOFLAG                                                           110
      LOGICAL READF                                                            120
      LOGICAL FIRSTF                                                           130
      LOGICAL PRINTF,SEQFLA,STAGEF                                             140
      INTEGER CONTIN                                                           150
      INTEGER WORDS(25,6)                                                      160
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                         170
      INTEGER TYPEPR                                                           180
      INTEGER CARDCN,OUTCNT                                                    190
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                       200
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                              210
      INTEGER WRDPTR                                                           220
      COMMON /SYMBOL/ WORDS                                                    230
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                      240
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN           250
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG          260
      COMMON PRINTF,SEQFLA,STAGEF                                              270
      COMMON FIRSTF                                                            280
      COMMON READF                                                             290
      COMMON EOFLAG                                                            300
C--------------                                                                310
C                                                                              320
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                     330
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                    340
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                   350
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                        360
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                   370
      EQUIVALENCE (CNTRS(1),WIS)                                               380
      EQUIVALENCE (CNTRS(2),SENTEN)                                            390
      EQUIVALENCE (CNTRS(3),PARAGR)                                            400
      EQUIVALENCE (CNTRS(4),CHAPTE)                                            410
      EQUIVALENCE (CNTRS(5),VOLUME)                                            420
      EQUIVALENCE (BANG,WORDS(20,4))                                           430
C                                                                              440
C     INIT COMMON.                                                             450
      WORDN=1                                                                  460
      VOLUME=0                                                                 470
      CHAPTE=1                                                                 480
      PARAGR=1                                                                 490
      SENTEN=1                                                                 500
      WIS=1                                                                    510
C     LIMIT OF DATA ON CARD.                                                   520
      LIMCOL=71                                                                530
      GO TO 1460                                                               540
C                                                                              550
C                                                                              560
1140  DO 1141 J1=1,6                                                           570
      IF(WORD(J1+1) .NE. WORDS(22,J1)) GO TO 1150                              580
1141  CONTINUE                                                                 590
```

```
C
C       NEW SERIES.
        WRDPTR=1
        CALL GETWRD
        IF(EOFLAG .EQ. 1) RETURN
        CALL CONVRT(WORD,WRDPTR,NUM)
        VOLUME=NUM
        GO TO 1461
C
1150    DO 1151 J1=1,6
        IF(WORD(J1+1) .NE. WORDS(23,J1)) GO TO 1160
1151    CONTINUE
C
C       NEW SESSION.
        WRDPTR=1
        CALL GETWRD
        CALL CONVRT(WORD,WRDPTR,NUM)
        CHAPTE=NUM
        GO TO 1462
C
1160    DO 1161 J1=1,6
        IF(WORD(J1+1) .NE. WORDS(24,J1)) GO TO 1531
1161    CONTINUE
C
C       NEW SPEAKER.
        WRDPTR=1
        CALL GETWRD
        CALL CONVRT(WORD,WRDPTR,NUM)
        PARAGR=NUM
        GO TO 1463
C
1460   VOLUME=VOLUME+1
1461    CHAPTE=1
1462    PARAGR=1
1463    SENTEN=1
1464    WIS=1
660     WRDPTR=1
        CALL GETWRD
        IF(EOFLAG .EQ. 1) RETURN
1430    IF(WORD(1) .NE. AMPERS) GO TO 1260
        IF(WORD(2) .NE. AMPERS) GO TO 1140
        IF(WORD(3) .NE. AMPERS) GO TO 1260
        IF(WRDPTR .EQ. 4) GO TO 1370
        IF(WORD(4) .NE. AMPERS) GO TO 1260
C       WORD = $$$$
C       WE HAVE END OF VOLUME NOW.
        GO TO 1460
C
C       WE HAVE END OF CHAPTER NOW.
C       WORD = $$$
1370    CHAPTE=CHAPTE+1
        GO TO 1462
1260    IF(WRDPTR .NE. 3) GO TO 1291
        IF(WORD(1) .NE. PERIOD) GO TO 1531
        IF(WORD(2) .NE. PERIOD) GO TO 1531
C       WORD = ..
C       WE HAVE END OF PARAGRAPH NOW.
        WRDPTR=2
        CALL WRTWRD
        PARAGR=PARAGR+1
```

```
      GO TO 1463                                                        1200
C                                                                       1210
1291  IF(WRDPTR .NE. 2) GO TO 1531                                      1220
      IF(WORD(1) .EQ. PERIOD) GO TO 1210                                1230
      IF(WORD(1) .EQ. QUESTM) GO TO 1210                                1240
      IF(WORD(1) .NE. BANG) GO TO 1531                                  1250
C     WE HAVE END OF SENTENCE NOW.                                      1260
1210  WRDPTR=2                                                          1270
      CALL WRTWRD                                                       1280
1220  SENTEN=SENTEN+1                                                   1290
      GO TO 1464                                                        1300
1531  CALL WRTWRD                                                       1310
      GO TO 660                                                         1320
      END                                                               1330
```

```
CPLAY         SUBROUTINE TO PROCESS -PLAY- TEXT.                                10
      SUBROUTINE PLAY                                                           20
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                            30
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                            40
      INTEGER AMPERS                                                            50
      INTEGER NUMBRS(10)                                                        60
      INTEGER STAR                                                              70
      INTEGER BANG                                                              80
C                                                                               90
C                                                                              100
C-------------------------------------------------------------------------     110
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.                 120
      INTEGER EOFLAG                                                           130
      LOGICAL READF                                                            140
      LOGICAL FIRSTF                                                           150
      LOGICAL PRINTF,SEQFLA,STAGEF                                             160
      INTEGER CONTIN                                                           170
      INTEGER WORDS(25,6)                                                      180
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                         190
      INTEGER TYPEPR                                                           200
      INTEGER CARDCN,OUTCNT                                                    210
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                       220
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                              230
      INTEGER WRDPTR                                                           240
      COMMON /SYMBOL/ WORDS                                                    250
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                      260
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN           270
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG          280
      COMMON PRINTF,SEQFLA,STAGEF                                              290
      COMMON FIRSTF                                                            300
      COMMON READF                                                             310
      COMMON EOFLAG                                                            320
C---------------                                                               330
C                                                                              340
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                     350
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                    360
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                   370
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                        380
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                   390
      EQUIVALENCE (CNTRS(1),WIS)                                               400
      EQUIVALENCE (CNTRS(2),SENTEN)                                            410
      EQUIVALENCE (CNTRS(3),PARAGR)                                            420
      EQUIVALENCE (CNTRS(4),CHAPTE)                                            430
      EQUIVALENCE (CNTRS(5),VOLUME)                                            440
      EQUIVALENCE (STAR,WORDS(11,6))                                           450
      EQUIVALENCE (BANG,WORDS(20,4))                                           460
C                                                                              470
      DATA NUMBRS/1H0,1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8,1H9/                     480
C     INIT COMMON.                                                             490
      WORDN=1                                                                  500
      VOLUME=1                                                                 510
      CHAPTE=1                                                                 520
      PARAGR=1                                                                 530
      SENTEN=1                                                                 540
C     IF WE ARE PROCESSING VERSE PLAY, THE FIRST READ WILL INCR                550
C     SENTEN COUNTER.                                                          560
      IF(TYPEPR .EQ. 5) SENTEN=0                                               570
C     LIMIT OF DATA ON CARD.                                                   580
      LIMCOL=71                                                                590
```

```
      GO TO 1464                                                        600
C                                                                       610
1600  VOLUME=VOLUME+1                                                   620
1461  CHAPTE=1                                                          630
1462  PARAGR=1                                                          640
1463  SENTEN=1                                                          650
1464  WIS=1                                                             660
C                                                                       670
660   WRDPTR=1                                                          680
662   CALL GETWRD                                                       690
      IF(EOFLAG .EQ. 1) RETURN                                          700
C     IS THIS A STAGE DIRECTION ---                                     710
      IF(WORD(1) .NE. STAR) GO TO 1429                                  720
C     YES, SHOULD IT BE IN OUTPUT DATASET ---                           730
      IF( .NOT. STAGEF) GO TO 662                                       740
      GO TO 1531                                                        750
C                                                                       760
C     ARE WE PROCESSING VERSE PLAY ---                                  770
1429  IF(TYPEPR .NE. 5) GO TO 1430                                      780
C     IS THIS A NEW LINE ---                                            790
      IF( .NOT. READF) GO TO 1430                                       800
      SENTEN=SENTEN+1                                                   810
      WIS=1                                                             820
1430  IF(WORD(1) .NE. AMPERS) GO TO 1260                                830
      IF(WORD(2) .NE. AMPERS) GO TO 1700                                840
      IF(WORD(3) .NE. AMPERS) GO TO 1531                                850
      IF(WORD(4) .EQ. AMPERS) GO TO 1600                                860
C     WORD=$$$                                                          870
C     WE HAVE END OF SCENE NOW.                                         880
1350  WRDPTR=1                                                          890
1370  CHAPTE=CHAPTE+1                                                   900
      GO TO 1462                                                        910
1260  IF(WORD(1) .NE. PERIOD) GO TO 1291                                920
      IF(WORD(2) .NE. PERIOD) GO TO 1291                                930
C     WORD = ..                                                         940
C     WE HAVE END OF PARAGRAPH NOW.                                     950
      WRDPTR=2                                                          960
      CALL WRTWRD                                                       970
1290  PARAGR=PARAGR+1                                                   980
      GO TO 1463                                                        990
C                                                                       1000
1700  DO 1701 J1=1,3                                                    1010
      IF(WORD(J1+1) .NE. WORDS(20,J1)) GO TO 1710                       1020
1701  CONTINUE                                                          1030
      WRDPTR=1                                                          1040
      CALL GETWRD                                                       1050
      IF(EOFLAG .EQ. 1) RETURN                                          1060
      CALL CONVRT(WORD,WRDPTR,NUM)                                      1070
C     RESET ACT.                                                        1080
      VOLUME=NUM                                                        1090
      GO TO 1461                                                        1100
1710  DO 1711 J1=1,5                                                    1110
      IF(WORD(J1+1) .NE. WORDS(21,J1)) GO TO 1531                       1120
1711  CONTINUE                                                          1130
      WRDPTR=1                                                          1140
      CALL GETWRD                                                       1150
      CALL CONVRT(WORD,WRDPTR,NUM)                                      1160
C     RESET SCENE.                                                      1170
      CHAPTE=NUM                                                        1180
      GO TO 1462                                                        1190
```

```
C                                                                       1200
1291    IF(WRDPTR .NE. 2) GO TO 1531                                    1210
        IF(WORD(1) .EQ. PERIOD) GO TO 1210                              1220
        IF(WORD(1) .EQ. QUESTM) GO TO 1210                              1230
        IF(WORD(1) .NE.  BANG ) GO TO 1531                              1240
C       WE HAVE END OF SENTENCE NOW.                                    1250
1210    WRDPTR=2                                                        1260
        CALL WRTWRD                                                     1270
1220    SENTEN=SENTEN+1                                                 1280
        GO TO 1464                                                      1290
1531    CALL WRTWRD                                                     1300
        GO TO 660                                                       1310
        END                                                             1320
```

```
CGETWRD     SUBROUTINE TO GET TEXT WORD FROM INPUT DATA.                    10
      SUBROUTINE GETWRD                                                     20
C     THIS IS GETWORD ROUTINE.                                              30
C     CANNOT LOCAL THIS SUBR.                                               40
C                                                                           50
C     P1 POINTS TO THE FIRST CHAR OF THE WORD IN THE TEXT STRING.           60
C     P2 POINTS TO THE LAST CHAR OF THE WORD IN THE TEXT STRING.            70
C     WRDPTR POINTS TO THE (LAST CHAR + 1) IN THE ,WORD, ARRAY.             80
      INTEGER P1,P2                                                         90
      INTEGER BLANKS,DASH                                                  100
      INTEGER ATSIGN                                                       110
C-------------------------------------------------------------------------120
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.            130
      INTEGER EOFLAG                                                       140
      LOGICAL READF                                                        150
      LOGICAL FIRSTF                                                       160
      LOGICAL PRINTF,SEQFLA,STAGEF                                         170
      INTEGER CONTIN                                                       180
      INTEGER WORDS(25,6)                                                  190
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                     200
      INTEGER TYPEPR                                                       210
      INTEGER CARDCN,OUTCNT                                                220
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                   230
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                          240
      INTEGER WRDPTR                                                       250
      COMMON /SYMBOL/ WORDS                                                260
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                  270
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN       280
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG      290
      COMMON PRINTF,SEQFLA,STAGEF                                          300
      COMMON FIRSTF                                                        310
      COMMON READF                                                         320
      COMMON EOFLAG                                                        330
C--------------                                                            340
C                                                                          350
      EQUIVALENCE (BLANKS,WORDS(9,6))                                      360
      EQUIVALENCE (DASH,WORDS(14,6))                                       370
      EQUIVALENCE (ATSIGN,WORDS(9,5))                                      380
C                                                                          390
      DATA P1/72/                                                          400
C                                                                          410
      READF=.FALSE.                                                        420
      P2=P1                                                                430
1780  IF(P1 .LE. LIMCOL) GO TO 1930                                        440
C     IS THIS WORD CONTINUED ---                                           450
      IF(CONTIN .EQ. ATSIGN) GO TO 1781                                    460
C     READ A NEW TEXT RECORD.                                              470
      CALL READIT                                                          480
      READF= .TRUE.                                                        490
1910  P1=1                                                                 500
1930  IF(CHAR(P1) .NE. BLANKS) GO TO 1979                                  510
1931  P1=P1+1                                                              520
      GO TO 1780                                                           530
C                                                                          540
1781  CALL READIT                                                          550
      GO TO 1910                                                           560
C                                                                          570
1979  P2=P1                                                                580
C     NOW WE HAVE FOUND START OF TEXT WORD.                                590
```

```
1980    P2=P2+1                                                          600
1990    IF(P2 .LE. LIMCOL) GO TO 2190                                    610
C       THE TEXT WORD IS IN COL 71.                                      620
2000    LIM =P2-1                                                        630
C       REMOVE WHEN COMMON VARIABLE WORD IS ENLARGED ********************640
        IF((WRDPTR+LIM-P1) .GT. 18) LIM=P1+18-WRDPTR                     650
        DO 2001 J1=P1,LIM                                                660
        WORD(WRDPTR)=CHAR(J1)                                            670
2001    WRDPTR=WRDPTR+1                                                  680
C       WRDPTR POINTS TO NEXT AVAILABLE CHARACTER IN WORD ARRAY.         690
C       IS THE TEXT WORD CONTINUED ---                                   700
2200    IF(CONTIN .EQ.DASH) GO TO 2050                                   710
        P1=P2+1                                                          720
999     RETURN                                                           730
C                                                                        740
C       THIS ROUTINE GETS A CONTINUED WORD FROM THE NEXT CARD.           750
2050    CALL READIT                                                      760
2170    P1=0                                                             770
        GO TO 1931                                                       780
C                                                                        790
2190    IF(CHAR(P2) .NE. BLANKS) GO TO 1980                              800
        LIM=P2-1                                                         810
C       REMOVE WHEN COMMON VARIABLE WORD IS ENLARGED ********************820
        IF((WRDPTR+LIM-P1) .GT. 18) LIM=P1+18-WRDPTR                     830
        DO 2300 J1=P1,LIM                                                840
        WORD(WRDPTR)=CHAR(J1)                                            850
2300    WRDPTR=WRDPTR+1                                                  860
        P1=P2+1                                                          870
        RETURN                                                           880
        END                                                              890
```

```
CREADIT     SUBR TO READ TEXT RECORD FROM INPUT DEVICE.                      10
      SUBROUTINE READIT                                                      20
C                                                                            30
      INTEGER PERIOD,BLANKS                                                  40
C                                                                            50
C-------------------------------------------------------------------------   60
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.              70
      INTEGER EOFLAG                                                         80
      LOGICAL READF                                                          90
      LOGICAL FIRSTF                                                        100
      LOGICAL PRINTF,SEQFLA,STAGEF                                          110
      INTEGER CONTIN                                                        120
      INTEGER WORDS(25,6)                                                   130
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                      140
      INTEGER TYPEPR                                                        150
      INTEGER CARDCN,OUTCNT                                                 160
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                    170
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                           180
      INTEGER WRDPTR                                                        190
      COMMON /SYMBOL/ WORDS                                                 200
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                   210
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN        220
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG       230
      COMMON PRINTF,SEQFLA,STAGEF                                           240
      COMMON FIRSTF                                                         250
      COMMON READF                                                          260
      COMMON EOFLAG                                                         270
C--------------                                                             280
C                                                                           290
      EQUIVALENCE (PERIOD,WORDS(14,1)), (BLANKS,WORDS(9,6))                 300
C                                                                           310
      READ(INFILE,9001)                                                     320
     *                  CHAR,CONTIN,PAGEN,SEQ                              330
9001  FORMAT(71A1,A1,I3,I5   )                                              340
C     CHECK FOR END-OF-FILE.                                                350
      IF(EOFLAG .EQ. 1) GO TO 999                                           360
C     INCREMENT CARD COUNT.                                                 370
      CARDCN=CARDCN+1                                                       380
C     IS SEQUENCE CHECKING REQUIRED ---                                     390
      IF( .NOT. SEQFLA) GO TO 1910                                          400
      IF(SEQ .GE. SEQOLD) GO TO 1890                                        410
      WRITE(SYSPRT,9002)                                                    420
9002  FORMAT(///,1H ,22HINPUT OUT OF SEQUENCE.,2X,                          430
     * 34HJOB TERMINATED AT THIS POINT *****)                               440
C     FORCE EOF CONDITION TO RETURN TO MAINLINE.                            450
      EOFLAG=1                                                              460
      GO TO 999                                                             470
C                                                                           480
1890  SEQOLD=SEQ                                                            490
1910  RETURN                                                                500
C                                                                           510
C     DUMMY DATA FROM EOF CONDITION.                                        520
C     MOVE PERIOD, BLANK.                                                   530
999   CHAR(1)=PERIOD                                                        540
      CHAR(2)=BLANKS                                                        550
      RETURN                                                                560
      END                                                                   570
```

```
COFLO       SUBR TO PRINT PAGE HEADINGS.                                      10
      SUBROUTINE OFLO                                                         20
C-------------------------------------------------------------------------  30
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.             40
      LOGICAL READF                                                         50
      LOGICAL FIRSTF                                                        60
      LOGICAL PRINTF,SEQFLA,STAGEF                                          70
      INTEGER CONTIN                                                        80
      INTEGER WORDS(25,6)                                                   90
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                     100
      INTEGER TYPEPR                                                       110
      INTEGER CARDCN,OUTCNT                                                120
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                   130
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                          140
      INTEGER WRDPTR                                                       150
      COMMON /SYMBOL/ WORDS                                                160
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                  170
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN       180
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG      190
      COMMON PRINTF,SEQFLA,STAGEF                                          200
      COMMON FIRSTF                                                        210
      COMMON READF                                                         220
C--------------                                                            230
10    WRITE(SYSPRT,9001) TITLE,IMON,IDAY,IYR,PAGE                          240
9001  FORMAT(1H1,20A4,19X,4HDATE,1X,2(A2,1H/),A2 ,5X,4HPAGE,I5,/)          250
C                                                                          260
      GO TO (999,100,200,300,400,500,600                          )        270
     * ,TYPEPR                                                             280
C                                                                          290
999   WRITE(SYSPRT,9000)                                                   300
9000  FORMAT(1H1,39HPROCESSING TYPE ERROR - JOB TERMINATED.)               310
      CALL EXIT                                                            320
C                                                                          330
100   WRITE(SYSPRT,9003)                                                   340
9003  FORMAT(1H ,23X,27H(VOLUME--CHAPTER--PARAGRAPH,                       350
     * 17H--SENTENCE--WORD),//)                                           360
      GO TO 998                                                            370
C                                                                          380
200   WRITE(SYSPRT,9004)                                                   390
9004  FORMAT(1H ,23X,30H(    --    --STANZA--LINE--WORD),//)               400
      GO TO 998                                                            410
C                                                                          420
300   WRITE(SYSPRT,9006)                                                   430
9006  FORMAT(1H ,23X,13H(ACT--SCENE--,                                     440
     * 26HPARAGRAPH--SENTENCE--WORD),//)                                   450
      GO TO 998                                                            460
C                                                                          470
400   WRITE(SYSPRT,9005)                                                   480
9005  FORMAT(1H ,23X,13H(ACT--SCENE--,                                     490
     * 22HPARAGRAPH--LINE--WORD),//)                                       500
      GO TO 998                                                            510
C                                                                          520
500   WRITE(SYSPRT,9002)                                                   530
9002  FORMAT(1H ,23X,40H(VOLUME--CHAPTER--PARAGRAPH--LINE--WORD),          540
     * //)                                                                 550
      GO TO 998                                                            560
C                                                                          570
600   WRITE(SYSPRT,9007)                                                   580
9007  FORMAT(1H ,23X,25H(SERIES--SESSION--SPEAKER,                         590
```

```
     * 17H--SENTENCE--WORD),//)                                    600
C                                                                  610
998     PAGE=PAGE+1                                                620
        LINES=4                                                    630
        RETURN                                                     640
        END                                                        650
```

```
CENDPAG     SUBR TO PRINT PAGE HEADING FOR ERROR MSGS.                      10
      SUBROUTINE ENDPAG                                                     20
C-----------------------------------------------------------------------    30
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.             40
      INTEGER EOFLAG                                                        50
      LOGICAL READF                                                         60
      LOGICAL FIRSTF                                                        70
      LOGICAL PRINTF,SEQFLA,STAGEF                                          80
      INTEGER CONTIN                                                        90
      INTEGER WORDS(25,6)                                                  100
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                     110
      INTEGER TYPEPR                                                       120
      INTEGER CARDCN,OUTCNT                                                130
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                   140
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                          150
      INTEGER WRDPTR                                                       160
      COMMON /SYMBOL/ WORDS                                                170
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                  180
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN       190
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG      200
      COMMON PRINTF,SEQFLA,STAGEF                                          210
      COMMON FIRSTF                                                        220
      COMMON READF                                                         230
      COMMON EOFLAG                                                        240
C--------------                                                            250
      WRITE(SYSPRT,9000) TITLE,                                            260
     *                   IMON,IDAY,IYR,PAGE                                270
9000  FORMAT(1H1,13HPROGRAM INDEX,4X,20A4,3X,                              280
     *                                6HDATE  ,2(A2,1H/),A2,5X,            290
     * 6HPAGE  ,I4)                                                        300
      WRITE(SYSPRT,9001)                                                   310
9001  FORMAT(, ,,,VERSION - MARCH 21, 1971.,,//)                           320
      PAGE=PAGE+1                                                          330
      LINES=3                                                              340
      FIRSTF=.FALSE.                                                       350
      RETURN                                                               360
      END                                                                  370
```

```
CINITCM     SUBROUTINE TO INITIALIZE BLANK COMMON.                          10
      SUBROUTINE INITCM                                                     20
C-----------------------------------------------------------------------    30
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.             40
      INTEGER EOFLAG                                                        50
      LOGICAL READF                                                         60
      LOGICAL FIRSTF                                                        70
      LOGICAL PRINTF,SEQFLA,STAGEF                                          80
      INTEGER CONTIN                                                        90
      INTEGER WORDS(25,6)                                                  100
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                     110
      INTEGER TYPEPR                                                       120
      INTEGER CARDCN,OUTCNT                                                130
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                   140
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                          150
      INTEGER WRDPTR                                                       160
      COMMON /SYMBOL/ WORDS                                                170
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                  180
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN       190
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG      200
      COMMON PRINTF,SEQFLA,STAGEF                                          210
      COMMON FIRSTF                                                        220
      COMMON READF                                                         230
      COMMON EOFLAG                                                        240
C-----------------------------------------------------------------------    250
C     PARAMETER CARD INPUT LOGICAL UNIT.                                   260
      SYSIN=5                                                              270
C     OUTPUT PRINT DEVICE LOGICAL UNIT.                                    280
      SYSPRT=6                                                             290
C     TEXT INPUT FILE LOGICAL UNIT.                                        300
      INFILE=20                                                            310
C     INDEX OUTPUT FILE LOGICAL UNIT.                                      320
      OUTFIL=21                                                            330
C     PAGE COUNTER.                                                        340
      PAGE=1                                                               350
C     PRINTED LINE COUNTER.                                                360
      LINES=0                                                              370
C     LINES PER PAGE.                                                      380
      LINEPG=60                                                            390
C     PRINT FLAG.                                                          400
      PRINTF=.FALSE.                                                       410
C     SEQUENCE CHECKING FLAG.                                              420
      SEQFLA=.FALSE.                                                       430
C     SEQUENCE FIELD FOR TESTING FIRST TEXT CARD.                          440
      SEQOLD=0                                                             450
C     STAGE DIRECTIONS FLAG.                                               460
      STAGEF=.FALSE.                                                       470
C     PAGE POSITIONING FLAG FOR ERROR MESSAGES.                            480
      FIRSTF=.TRUE.                                                        490
C     INDICATOR FOR NEW RECORD READ.                                       500
      READF=.FALSE.                                                        510
C     TYPE OF PROCESSING.                                                  520
      TYPEPR=1                                                             530
C     PTR TO NEXT AVAIL CHAR IN OUTPUT WORD ARRAY.                         540
      WRDPTR=1                                                             550
C     COUNT OF INPUT TEXT RECORDS.                                         560
      CARDCN=0                                                             570
C     COUNT OF OUTPUT INDEX RECORDS.                                       580
      OUTCNT=0                                                             590
```

```
C       COLUMN DATA LIMIT.                                              600
        LIMCOL=71                                                       610
        RETURN                                                          620
        END                                                             630
```

```
CSYMBOL     SUBPROGRAM TO INITIALIZE LABELED COMMON.                          10
      BLOCK DATA                                                              20
C-----------------------------------------------------------------------      30
      INTEGER WORDS(25,6)                                                     40
      COMMON /SYMBOL/ WORDS                                                   50
C-----------------------------------------------------------------------      60
      DATA (WORDS(1,J),J=1,6)/1HP,1HR,1HO,1HC,1H=,1H /                        70
      DATA (WORDS(2,J),J=1,6)/1HP,1HR,1HO,1HS,1H,,1H /                        80
      DATA (WORDS(3,J),J=1,6)/1HP,1HO,1HE,1HT,1H,,1H /                        90
      DATA (WORDS(4,J),J=1,6)/1HP,1HL,1HA,1HY,1H,,1H /                       100
      DATA (WORDS(5,J),J=1,6)/1HV,1HP,1HL,1HA,1H,,1H /                       110
      DATA (WORDS(6,J),J=1,6)/1HM,1HI,1HL,1HT,1H,,1H /                       120
      DATA (WORDS(8,J),J=1,6)/1HP,1HR,1HI,1HN,1HT,1H=/                       130
C     AMPERSAND - WORDS(7,6)                                                 140
      DATA (WORDS(7,J),J=1,6)/1HS,1HP,1HO,1HK,1H,,1H+/                       150
C     WORDS(9,4) MUST BE BLANK.                                              160
C     ATSIGN - WORDS(9,5)                                                    170
C     BLANKS - WORDS(9,6)                                                    180
      DATA (WORDS(9,J),J=1,6)/1HN,1HO,1H,,1H ,1H,,1H /                       190
      DATA (WORDS(10,J),J=1,6)/1HY,1HE,1HS,1H,,2*1H /                        200
C     STAR - WORDS(11,6)                                                     210
      DATA (WORDS(11,J),J=1,6)/1HS,1HE,1HQ,1H=,1H ,1H*/                      220
      DATA (WORDS(12,J),J=1,6)/1HS,1HT,1HA,1HG,1HE,1H=/                      230
      DATA (WORDS(13,J),J=1,6)/1H+,1HV,1HO,1HL,1HU,1HM/                      240
C     WORDS(14,J) PERIOD,QUESTM,APOSTR,QUOTE,DOLLAR,DASH                     250
      DATA (WORDS(14,J),J=1,6)/1H.,0172020202020,0572020202020,              260
     * 0762020202020,1H$,1H-/                                                270
      DATA (WORDS(15,J),J=1,6)/1HS,1HE,1HN,1HT,1HE,1HN/                      280
      DATA (WORDS(16,J),J=1,6)/1HP,1HA,1HR,1HA,1HG,1HR/                      290
      DATA (WORDS(17,J),J=1,6)/1HC,1HH,1HA,1HP,1HT,1HE/                      300
      DATA (WORDS(18,J),J=1,6)/1HV,1HO,1HL,1HU,1HM,1HE/                      310
C     WORDS(19,J) AVAILABLE FOR FUTURE.                                      320
C     BANG (EXCLAMATION) - WORDS(20,4)                                       330
      DATA (WORDS(20,J),J=1,6)/1HA,1HC,1HT,0772020202020,1H ,1H /            340
      DATA (WORDS(21,J),J=1,6)/1HS,1HC,1HE,1HN,1HE,1HZ/                      350
      DATA (WORDS(22,J),J=1,6)/1HS,1HE,1HR,1HI,1HE,1HS/                      360
      DATA (WORDS(23,J),J=1,6)/1HS,1HE,1HS,1HS,1HI,1HO/                      370
      DATA (WORDS(24,J),J=1,6)/1HS,1HP,1HE,1HA,1HK,1HE/                      380
      DATA(WORDS(25,J),J=1,6)/1HD,1HE,1HL,1HI,1HM,1H=/                       390
      END                                                                    400
```

```
CPOET        SUBROUTINE TO PROCESS -POET- TEXT.                                 10
      SUBROUTINE POET                                                           20
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                            30
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                            40
      INTEGER AMPERS                                                            50
      INTEGER STANZA                                                            60
      INTEGER BANG                                                              70
C                                                                               80
C                                                                               90
C-----------------------------------------------------------------------       100
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.                 110
      INTEGER EOFLAG                                                            120
      LOGICAL READF                                                             130
      LOGICAL FIRSTF                                                            140
      LOGICAL PRINTF,SEQFLA,STAGEF                                              150
      INTEGER CONTIN                                                            160
      INTEGER WORDS(25,6)                                                       170
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                          180
      INTEGER TYPEPR                                                            190
      INTEGER CARDCN,OUTCNT                                                     200
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                        210
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                               220
      INTEGER WRDPTR                                                            230
      COMMON /SYMBOL/ WORDS                                                     240
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                       250
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN            260
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG           270
      COMMON PRINTF,SEQFLA,STAGEF                                               280
      COMMON FIRSTF                                                             290
      COMMON READF                                                              300
      COMMON EOFLAG                                                             310
C--------------                                                                 320
C                                                                               330
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                      340
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                     350
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                    360
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                         370
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                    380
      EQUIVALENCE (CNTRS(1),WIS)                                                390
      EQUIVALENCE (CNTRS(2),SENTEN)                                             400
      EQUIVALENCE (CNTRS(3),PARAGR)                                             410
      EQUIVALENCE (CNTRS(4),CHAPTE)                                             420
      EQUIVALENCE (CNTRS(5),VOLUME)                                             430
      EQUIVALENCE (STANZA,PARAGR)                                               440
      EQUIVALENCE (BANG,WORDS(20,4))                                            450
C                                                                               460
C     INIT COMMON.                                                              470
      LIMCOL=71                                                                 480
      WORDN=1                                                                   490
      VOLUME=0                                                                  500
      CHAPTE=1                                                                  510
      PARAGR=1                                                                  520
      SENTEN=0                                                                  530
      GO TO 1464                                                                540
C                                                                               550
1463  SENTEN=1                                                                  560
1464  WIS=1                                                                     570
660   WRDPTR=1                                                                  580
      CALL GETWRD                                                               590
```

```
      IF(EOFLAG .EQ. 1) RETURN                                          600
      IF(PAGEN .EQ. STANZA) GO TO 1250                                  610
      STANZA=PAGEN                                                      620
      SENTEN=1                                                          630
      WIS=1                                                             640
      GO TO 1260                                                        650
C                                                                       660
1250  IF( .NOT. READF) GO TO 1260                                       670
      SENTEN=SENTEN+1                                                   680
      WIS=1                                                             690
C                                                                       700
C                                                                       710
1260  IF(WRDPTR .NE. 3) GO TO 1291                                      720
      IF(WORD(1) .NE. PERIOD) GO TO 1531                                730
      IF(WORD(2) .NE. PERIOD) GO TO 1531                                740
C     WORD = ..                                                         750
C     WE HAVE END OF STANZA NOW.                                        760
      WRDPTR=2                                                          770
      CALL WRTWRD                                                       780
      STANZA=STANZA+1                                                   790
      GO TO 1463                                                        800
C                                                                       810
1291  IF(WRDPTR .NE. 2) GO TO 1531                                      820
      IF(WORD(1) .EQ. PERIOD) GO TO 1210                                830
      IF(WORD(1) .EQ. QUESTM) GO TO 1210                                840
      IF(WORD(1) .NE.  BANG ) GO TO 1531                                850
C     WE HAVE END OF SENTENCE NOW.                                      860
1210  WRDPTR=2                                                          870
      CALL WRTWRD                                                       880
1220  SENTEN=SENTEN+1                                                   890
      GO TO 1464                                                        900
1531  CALL WRTWRD                                                       910
      GO TO 660                                                         920
      END                                                               930
```

```
CMILT       SUBROUTINE TO PROCESS ,MILT, TEXT.                                940
      SUBROUTINE MILT                                                         950
      INTEGER CHAPTE,PARAGR,SENTEN,WIS,VOLUME,BLANKS                          960
      INTEGER PERIOD,DOLLAR,QUESTM,DASH,QUOTE,APOSTR                          970
      INTEGER AMPERS                                                          980
      INTEGER ATSIGN                                                          990
      INTEGER BANG                                                           1000
C                                                                            1010
C                                                                            1020
C-------------------------------------------------------------------------   1030
C     THIS GROUP OF CARDS GOES IN ALL SUBROUTINES AND MAINLINE.              1040
      INTEGER EOFLAG                                                         1050
      LOGICAL READF                                                          1060
      LOGICAL FIRSTF                                                         1070
      LOGICAL PRINTF,SEQFLA,STAGEF                                           1080
      INTEGER CONTIN                                                         1090
      INTEGER WORDS(25,6)                                                    1100
      INTEGER SYSPRT,SYSIN,OUTFIL,PAGE                                       1110
      INTEGER TYPEPR                                                         1120
      INTEGER CARDCN,OUTCNT                                                  1130
      INTEGER TITLE(20),CHAR(71),CONTIN,PAGEN,SEQ,SEQOLD                     1140
      INTEGER CNTRS(6),WORDN,WORDL,PAGEN,WORD(18)                            1150
      INTEGER WRDPTR                                                         1160
      COMMON /SYMBOL/ WORDS                                                  1170
      COMMON SYSPRT,SYSIN,INFILE,OUTFIL,PAGE,LINES,TYPEPR                    1180
      COMMON SEQ,SEQOLD,LIMCOL,CHAR,TITLE,PAGEN,CONTIN,WRDPTR,CARDCN         1190
      COMMON CNTRS,WORDN,WORDL,      WORD,OUTCNT,IDAY,IMON,IYR,LINEPG        1200
      COMMON PRINTF,SEQFLA,STAGEF                                            1210
      COMMON FIRSTF                                                          1220
      COMMON READF                                                           1230
      COMMON EOFLAG                                                          1240
C--------------                                                              1250
C                                                                            1260
      EQUIVALENCE (DOLLAR,WORDS(14,5)), (DASH,WORDS(14,6))                   1270
      EQUIVALENCE (QUOTE,WORDS(14,4)), (APOSTR,WORDS(14,3))                  1280
      EQUIVALENCE (PERIOD,WORDS(14,1)), (QUESTM,WORDS(14,2))                 1290
      EQUIVALENCE (BLANKS,WORDS(9,6)  )                                      1300
      EQUIVALENCE (AMPERS,WORDS(  7,6     ))                                 1310
      EQUIVALENCE (CNTRS(1),WIS)                                             1320
      EQUIVALENCE (CNTRS(2),SENTEN)                                          1330
      EQUIVALENCE (CNTRS(3),PARAGR)                                          1340
      EQUIVALENCE (CNTRS(4),CHAPTE)                                          1350
      EQUIVALENCE (CNTRS(5),VOLUME)                                          1360
      EQUIVALENCE (ATSIGN,WORDS(9,5))                                        1370
      EQUIVALENCE (BANG,WORDS(20,4))                                         1380
C                                                                            1390
C     INIT COMMON.                                                           1400
C     LIMIT OF DATA ON CARD.                                                 1410
      LIMCOL=71                                                              1420
      WORDN=1                                                                1430
      VOLUME=1                                                               1440
      CHAPTE=1                                                               1450
      PARAGR=1                                                               1460
      SENTEN=0                                                               1470
      GO TO 1464                                                             1480
C                                                                            1490
1462  PARAGR=1                                                               1500
1463  SENTEN=1                                                               1510
1464  WIS=1                                                                  1520
```

```
C
C
660     WRDPTR=1                                                        1530
        CALL GETWRD                                                     1540
        IF(EOFLAG .EQ. 1) RETURN                                        1550
C       VOLUME NUMBER IN COLS 73-74.                                    1560
        IF(PAGEN .EQ. VOLUME) GO TO 1429                                1570
        VOLUME=PAGEN                                                    1580
        CHAPTE=1                                                        1590
        PARAGR=1                                                        1600
        SENTEN=1                                                        1610
        WIS=1                                                           1620
        GO TO 1430                                                      1630
C                                                                       1640
1429    IF( .NOT. READF) GO TO 1430                                     1650
        SENTEN=SENTEN+1                                                 1660
        WIS=1                                                           1670
C                                                                       1680
1430    IF(WRDPTR .GT. 4) GO TO 1531                                    1690
        IF(WORD(1) .NE. AMPERS) GO TO 1260                              1700
        IF(WORD(2) .NE. AMPERS) GO TO 1260                              1710
        IF(WORD(3) .NE. AMPERS) GO TO 1260                              1720
C                                                                       1730
C       WE HAVE END OF CHAPTER NOW.                                     1740
C       WORD = $$$                                                      1750
1350    WRDPTR=1                                                        1760
1370    CHAPTE=CHAPTE+1                                                 1770
        GO TO 1462                                                      1780
1260    IF(WRDPTR .NE. 3) GO TO 1291                                    1790
        IF(WORD(1) .NE. PERIOD) GO TO 1531                              1800
        IF(WORD(2) .NE. PERIOD) GO TO 1531                              1810
C       WORD = ..                                                       1820
C       WE HAVE END OF PARAGRAPH NOW.                                   1830
        WRDPTR=2                                                        1840
        CALL WRTWRD                                                     1850
1290    PARAGR=PARAGR+1                                                 1860
        GO TO 1463                                                      1870
C                                                                       1880
1291    IF(WRDPTR .NE. 2) GO TO 1531                                    1890
        IF(WORD(1) .EQ. PERIOD) GO TO 1210                              1900
        IF(WORD(1) .EQ. QUESTM) GO TO 1210                              1910
        IF(WORD(1) .NE. BANG) GO TO 1531                                1920
C                                                                       1930
C       WE HAVE END OF SENTENCE NOW.                                    1940
1210    WRDPTR=2                                                        1950
        CALL WRTWRD                                                     1960
1220    SENTEN=SENTEN+1                                                 1970
        GO TO 1464                                                      1980
1531    CALL WRTWRD                                                     1990
        GO TO 660                                                       2000
        END                                                             2010
                                                                        2020
                                                                        2030
```

```
CSUFNA1        BUILD FUNCTION RECORDS-LOGIC
C
C             PROGRAM SUFNA1 IS USED TO CREATE A FILE OF
C       FUNCTION WORDS AND PUNCTUATION SYMBOLS FOR USE BY THE SUFFIX
C       PROGRAM.  THE INPUT CONSISTS OF 80 CHARACTER CARD IMAGES.  THE
C       FUNCTION WORD OR PUNCTUATION SYMBOL MUST BEGIN IN COLUMN 1 AND
C       MUST BE FOLLOWED BY AT LEAST ONE BLANK.  THE RECORD MUST ALSO
C       CONTAIN EITHER THE WORD ,FUNCTION, OR THE WORD ,PUNCTUATION,,
C       DEPENDING ON THE TYPE OF RECORD.  AFTER THE FILE HAS BEEN BUILT
C       IT SHOULD BE SORTED INTO COLLATING SEQUENCE ORDER (IF THE
C       INPUT RECORDS WERE NOT ALREADY SO SORTED).
C             THE FUNCTION WORD AND PUNCTUATION FILE IS READ IN ON
C       FORTRAN LOGICAL UNIT 10
C             THE OUTPUT FILE IS WRITTEN IN BINARY (I.E. UNFORMATTED--
C       NO CHANGE IS MADE IN THE FORMAT OF THE RECORD) TO LOGICAL UNIT
C       35
C             THE AUDIT LISTING IS WRITTEN TO LOGICAL UNIT 6
C
      DIMENSION KARD(80),IWORD(18)
      DATA KCURR,IFUNCT,IPUNCT,NRJECT,IOUTCT,LINES,LINERR,LABEL/8*0/,
     1  IBLANK/6H      /,NOCHAR/18/
      DATA (IWORD(J),J=1,18)/18*6H      /
      DATA IF/1HF/,IP/1HP/
      DATA LEOF/0772020202020/
      DATA LIMPAG/60/
C
C
C       GO TO WRITE HEADING FOR FIRST PAGE OF AUDIT REPORT.
C       LABEL IS USED AS A SWITCH TO CAUSE RETURN OF CONTROL TO
C       STATEMENT 20200
      LABEL = 1
      GO TO 40000
C
C       INCOUN KEEPS TRACK OF RECORDS READ IN.
20200 INCOUN = 0
C
C       READ FUNCTION RECORD AND CHECK FOR END OF FILE.
C
20300 READ(10,20350) KARD
20350 FORMAT (80A1)
      IF(KARD(1).EQ.LEOF.AND.KARD(2).EQ.LEOF.AND.KARD(3).EQ.LEOF)
     1  GO TO 50000
20400 I = 1
      INCOUN = INCOUN + 1
C
C       THE FUNCTION WORD OR PUNCTUATION SYMBOL MUST BEGIN IN COLUMN 1.
C
      IF(KARD(1).EQ.IBLANK) GO TO 71000
20500 IF(I.GE.NOCHAR) GO TO 72000
C
C       MOVE CHARACTER TO HOLD AREA.
      IWORD(I) = KARD(I)
      I = I + 1
      IF(KARD(I).NE.IBLANK) GO TO 20500
C
C       INPUT WORD HAS BEEN MOVED TO SAVE AREA.  CHECK FOR PUNCTUATION
C        OR FUNCTION.
      J = I - 1
20600 I = I + 1
```

```
      IF(KARD(I).NE.IBLANK) GO TO 20700
      IF(I.GE.80) GO TO 73000
      GO TO 20600
20700 IF(KARD(I).NE.IF) GO TO 20800
C
C       RECORD IS A FUNCTION WORD RECORD.
C
      ITYPE = 2
      GO TO 20900
20800 IF(KARD(I).NE.IP) GO TO 74000
C
C       RECORD IS A PUNCTUATION RECORD.
C
      ITYPE = 1
20900 K = 1
21000 I = I + 1
      IF(KARD(I).EQ.IBLANK) GO TO 21100
      K = K + 1
      IF(I.LT.80) GO TO 21000
C
C       CHECK FOR PROPER NUMBER OF LETTERS AND UPDATE TYPE COUNTER FOR
C        AUDIT REPORT.
C
21100 GO TO (21200,21300), ITYPE
21200 IF(K.NE.11) GO TO 74000
      IPUNCT = IPUNCT + 1
      GO TO 21400
21300 IF(K.NE.8) GO TO 74000
      IFUNCT = IFUNCT + 1
C
C       UPDATE OUTPUT RECORD COUNTER AND WRITE RECORD TO FUNCTION WORD
C        FILE.
21400 IOUTCT = IOUTCT + 1
      WRITE (35) ITYPE,J,IWORD
        LINES = LINES + 1
C
C       CHECK NUMBER OF LINES IN AUDIT REPORT.
      IF(LINES.LT.LIMPAG) GO TO 21500
      LABEL = 2
      GO TO 40000
C
C       WRITE AUDIT RECORD.
21500 WRITE (6,21550) ITYPE,IWORD
21550 FORMAT(3X,I2,5X,18A1)
      LINES = LINES + 1
C
C       BLANK OUT SAVE AREA.
      DO 21600 L=1,J
      IWORD(L) = IBLANK
21600 CONTINUE
      GO TO 20300
C
C       NEW PAGE ROUTINE.
C
40000 NPAGE = NPAGE + 1
      WRITE (6,40100) NPAGE
40100 FORMAT (1H1,20X,49HFUNCTION WORDS (TYPE=2) AND PUNCTUATION (TYPE=1
     1), 10X,5HPAGE ,I5//2X,4HTYPE,5X,4HWORD//)
      LINES = 4
      GO TO (20200,21500), LABEL
```

```
C
C       END OF FILE.  SUMMARIZE AUDIT REPORT.
C
50000 WRITE (6,40100) NPAGE
      WRITE (6,50100) INCOUN,IOUTCT,NRJECT,IFUNCT,IPUNCT
50100 FORMAT (////,15X,28H* * *  END OF LISTING  * * */1HO,5X,18HRECORDS
     1 READ IN-- ,I5/6X,22HRECORDS WRITTEN OUT-- ,I5/6X,19HRECORDS REJEC
     2TED// ,I5/6X,24HFUNCTION WORD RECORDS-- ,I5/6X,22HPUNCTUATION RECO
     3RDS-- ,I5)
      ITYPE = 0
      WRITE (35) ITYPE,J,IWORD
      REWIND 35
      STOP
C
C       ERROR MESSAGES.
C
71000 WRITE(6,71100)
71100 FORMAT (1X61H*** INVALID RECORD FORMAT, BLANK IN COLUMN 1.  CARD R
     1EJECTED.)
      GO TO 79000
72000 WRITE (6,72100) NOCHAR
72100 FORMAT (1X,51H*** INVALID RECORD FORMAT, WORD CONTAINS MORE THAN ,
     1 I3,28H CHARACTERS.  CARD REJECTED.)
      GO TO 79000
73000 WRITE (6,73100)
73100 FORMAT (1X,65H*** INVALID RECORD FORMAT, TYPE FIELD IS MISSING.  C
     1ARD REJECTED.)
      GO TO 79000
74000 WRITE (6,74100)
74100 FORMAT (1X,67H*** INVALID RECORD FORMAT, TYPE FIELD IS INCORRECT.
     1 CARD REJECTED.)
79000 WRITE (6,79100) KARD
79100 FORMAT (15X,8HCARD = ,,80A1,1H,)
      NRJECT = NRJECT + 1
      LINERR = LINERR + 2
      IF(LINERR.LT.LIMPAG) GO TO 20300
      WRITE (6,79200)
79200 FORMAT (1H1)
      LINERR = 1
      GO TO 20300
      END
```

```
CSUFN1S        READ SUFFIX RECORDS-LOGIC
C             THIS PROGRAM READS IN SUFFIX PAIRS AND ASSOCIATED
C       EXCEPTION WORDS AND LETTERS.  THE FIRST SUFFIX MUST BEGIN IN
C       COLUMN 1, IF COLUMN 1 IS BLANK THE FIRST SUFFIX IS THE BLANK.
C       FOLLOWING THE FIRST SUFFIX, THERE MUST BE AT LEAST ONE BLANK
C       AND THEN THE SECOND SUFFIX.
C             THE CARD CONTAINING THE SUFFIX PAIR SHOULD THEN BE FOLLOW-
C       ED BY ALL ASSOCIATED EXCEPTION RECORDS.  THE WORD ,EXCEPTION,
C       MUST BEGIN IN COLUMN 1, FOLLOWED BY AT LEAST 1 BLANK.  IF THE
C       RECORD IS FOR AN EXCEPTION LETTER, THEN THE WORD ,LETTER, IS
C       TYPED FOLLOWED BY AT LEAST ONE BLANK AND THEN THE EXCEPTION
C       LETTER.
C
C
      DIMENSION ISUFF1(18),ISUFF2(18),IEXCEP(18),ITEMP(18)
      DIMENSION KARD(80),LETTR(7),MXCPT(6)
      DATA (ISUFF1(K),ISUFF2(K),IEXCEP(K),K=1,18),NOCHAR/54*6H      ,18/
      DATA IBLANK,KE,KL,(LETTR(K),K=1,7),(MXCPT(K),K=1,6)/6H      ,1HE,
     1 2*1HL,1HE,1HT,1HT,1HE,1HR,1H ,1HX,1HC,1HE,1HP,1HT,1H /,INCOUN,
     2  NRJECT,IOTCNT/3*0/
      DATA LEOF/0772020202020/
      DATA LIMPAG/60/
C
C
 9900 L = 0
C       J1 IS USED TO KEEP TRACK OF NUMBER OF CHARACTERS IN FIRST SUFFIX
      J1 = 1
      I = 1
C       READ IN CARD.
      READ(10,10000) KARD
10000 FORMAT (80A1)
      IF(KARD(1).EQ.LEOF.AND.KARD(2).EQ.LEOF.AND.KARD(3).EQ.LEOF)
     1  GO TO 80000
      INCOUN = INCOUN + 1
C       IF FIRST COLUMN IS BLANK, THEN FIRST SUFFIX IS THE BLANK.
10100 IF(KARD(I).EQ.IBLANK) GO TO 10500
C       BUILD SUFFIX IN SAVE AREA.
10200 ISUFF1(I) = KARD(I)
      I = I + 1
      IF(KARD(I).EQ.IBLANK) GO TO 10300
      IF(I.GT.NOCHAR) GO TO 71000
      GO TO 10200
10300 J1 = I - 1
      IF(L.NE.0) GO TO 10500
      L = 1
      IF(I.NE.7) GO TO 10500
      IF(KARD(1).NE.KE) GO TO 10500
      DO 10400 K=1,5
      IF(KARD(K+1).NE.MXCPT(K)) GO TO 10500
10400 CONTINUE
      GO TO 72000
C       J2 IS USED TO KEEP TRACK FO NUMBER OF CHARACTERS IN SECOND
C         SUFFIX.
10500 J2 = 0
10600 I = I + 1
C       FIND THE BEGINNING OF THE SECOND SUFFIX.
      IF(KARD(I).NE.IBLANK) GO TO 10700
      IF(I.GE.80) GO TO 73000
      GO TO 10600
```

```
10700 J2 = J2 + 1
C       BUILD SECOND SUFFIX IN SAVE AREA.
      ISUFF2(J2) = KARD(I)
      IF(I.GE.80) GO TO 10800
      I = I + 1
      IF(KARD(I).EQ.IBLANK) GO TO 10800
      IF(J2.GE.NOCHAR) GO TO 74000
      GO TO 10700
10800 ITYPE = IBLANK
C       THE SUFFIX THAT IS LOWER IN COLLATING SEQUENCE SHOULD BE THE
C         FIRST MEMBER OF THE PAIR.
      DO 10820 L=1,NOCHAR
      LTEST = LCOMP(ISUFF1(L),ISUFF2(L))
      IF(LTEST) 10880,10810,10840
10810 IF(ISUFF1(L).EQ.IBLANK) GO TO 10880
10820 CONTINUE
      GO TO 10880
C       MAKE THE INTERCHANGE
10840 DO 10860 L=1,NOCHAR
      ITEMP(L) = ISUFF1(L)
      ISUFF1(L) = ISUFF2(L)
      ISUFF2(L) = ITEMP(L)
10860 CONTINUE
      JHOLD = J1
      J1 = J2
      J2 = JHOLD
C       WRITE OUT SUFFIX RECORD
10880 WRITE (20)        ISUFF1,ISUFF2,ITYPE,IEXCEP
      IOTCNT = IOTCNT + 1
10900 J = 1
      I = 1
C       READ IN NEXT RECORD.
      READ(10,10000) KARD
      IF(KARD(1).EQ.LEOF.AND.KARD(2).EQ.LEOF.AND.KARD(3).EQ.LEOF)
     1  GO TO 80000
      INCOUN = INCOUN + 1
C       IS THE RECORD AN EXCEPTION RECORD.
      IF(KARD(1).NE.KE) GO TO 11100
11000 I = I + 1
      IF(KARD(I).EQ.MXCPT(J)) GO TO 11050
      I = 1
      GO TO 11100
11050 IF(I.GT.6) GO TO 11700
      J = J + 1
      GO TO 11000
C
C       RECORD IS NOT AN EXCEPTION RECORD.  CLEAR SAVE AREAS.
11100 DO 11200 K=1,J1
      ISUFF1(K) = IBLANK
11200 CONTINUE
      DO 11300 K=1,J2
      ISUFF2(K) = IBLANK
11300 CONTINUE
C       IF FIRST COLUMN IS BLANK THEN THE FIRST SUFFIX IS THE BLANK
      IF(KARD(I).NE.IBLANK) GO TO 11400
      J1 = 1
      GO TO 10500
11400 IF(I.GE.NOCHAR) GO TO 71000
      I = I + 1
      IF(KARD(I).NE.IBLANK) GO TO 11400
```

```
11500 J1 = I - 1
C       BUILD FIRST SUFFIX IN SAVE AREA.
      DO 11600 K=1,J1
      ISUFF1(K) = KARD(K)
11600 CONTINUE
C       GO TO FIND SECOND SUFFIX.
      GO TO 10500
C
C       RECORD IS EXCEPTION RECORD.
C
11700 ITYPE = KE
11800 I = I + 1
      IF(KARD(I).NE.IBLANK) GO TO 11900
      IF(I.GE.80) GO TO 75000
      GO TO 11800
11900 J3 = 1
C       IS THIS AN EXCEPTION LETTER RECORD.
12000 IF(KARD(I).NE.LETTR(J3)) GO TO 12300
      IF(J3.GE.7) GO TO 12100
      IF(I.GE.80) GO TO 12600
      J3 = J3 + 1
      I = I + 1
      GO TO 12000
C
C       THIS IS AN EXCEPTION LETTER RECORD.
C
12100 ITYPE = KL
12200 IF(I.GE.80) GO TO 76000
      I = I + 1
C       FIND LETTER.
      IF(KARD(I).EQ.IBLANK) GO TO 12200
      IF(KARD(I+1).NE.IBLANK) GO TO 77000
      J3 = 1
      IEXCEP(1) = KARD(I)
C       GO TO WRITE OUT RECORD.
      GO TO 12800
C
C       THIS IS AN EXCEPTION WORD RECORD.
C
12300 IF(KARD(I).NE.IBLANK) GO TO 12400
      J3 = J3 - 1
      GO TO 12600
12400 IF(I.GE.80) GO TO 12600
      I = I + 1
C       FIND END OF EXCEPTION RECORD.
      IF(KARD(I).EQ.IBLANK) GO TO 12600
      J3 = J3 + 1
      GO TO 12400
12600 J4 = I - J3 - 1
C       BUILD EXCEPTION WORD IN SAVE AREA.
      DO 12700 K=1,J3
      K1 = K + J4
      IEXCEP(K) = KARD(K1)
12700 CONTINUE
C       WRITE OUT EXCEPTION RECORD.
12800 WRITE (20)        ISUFF1,ISUFF2,ITYPE,IEXCEP
      IOTCNT = IOTCNT + 1
      DO 12900 K = 1,J3
      IEXCEP(K) = IBLANK
12900 CONTINUE
```

```
      GO TO 10900
C
C       ERROR MESSAGES.
C
71000 WRITE (6,71100)
71100 FORMAT (1X,66H* * * * TOO MANY CHARACTERS IN FIRST SUFFIX FIELD.
     1CARD REJECTED.)
      JUMP = 1
      GO TO 79000
72000 WRITE (6,72100)
72100 FORMAT (1X,55H* * * * NO SUFFIX FOR EXCEPTION RECORD.  CARD REJECT
     1ED.)
      L = 0
      JUMP = 1
      GO TO 79000
73000 WRITE (6,73100)
73100 FORMAT (1X,46H* * * * SECOND SUFFIX MISSING.  CARD REJECTED.)
      L = 0
      JUMP = 1
      GO TO 79000
74000 WRITE (6,74100)
74100 FORMAT (1X,67H* * * * TOO MANY CHARACTERS IN SECOND SUFFIX FIELD.
     1 CARD REJECTED.)
      L = 0
      JUMP = 1
      GO TO 79000
75000 WRITE (6,75100)
75100 FORMAT(1X,47H* * * * EXCEPTION WORD MISSING.  CARD REJECTED.)
      JUMP = 2
      GO TO 79000
76000 WRITE (6,76100)
76100 FORMAT(1X,49H* * * * EXCEPTION LETTER MISSING.  CARD REJECTED.)
      JUMP = 2
      GO TO 79000
77000 WRITE (6,77100)
77100 FORMAT (1X,65H* * * * EXCEPTION LETTER HAS TOO MANY CHARACTERS.  C
     1ARD REJECTED.)
      JUMP = 2
79000 WRITE (6,79100) KARD
79100 FORMAT (15X,8HCARD = ,,80A1,1H,)
      NRJECT = NRJECT + 1
      LINERR = LINERR + 2
      INCOUN = INCOUN - 1
      IF(LINERR.LT.LIMPAG) GO TO 79300
      WRITE (6,79200)
79200 FORMAT (1H1)
      LINERR = 1
79300 GO TO (9900,10900), JUMP
80000 WRITE (6,80100) INCOUN,IOTCNT,NRJECT
80100 FORMAT (1H1,1X,27HEND OF STEP SUFUNA, PART 1.///10X,17HRECORDS REA
     1D IN--,I5/10X,21HRECORDS WRITTEN OUT--,I5/10X,18HRECORDS REJECTED-
     2-,I5)
      DO 80200 KI=1,18
      ISUFF1(KI) = LEOF
80200 CONTINUE
      WRITE (20) ISUFF1,ISUFF2,ITYPE,IEXCEP
      REWIND 20
      STOP
      END
```

```
CSUFNS2        BUILD SUFFIX FILE-LOGIC
C
C             PROGRAM SUFNS2 ACCEPTS THE SORTED FILE OF SUFFIX PAIRS AND
C       ASSOCIATED EXCEPTION WORDS AND LETTERS WHICH WAS CREATED BY
C       SUFN1S.  IT BUILDS TABLES OF SUFFIXES, EXCEPTION WORDS AND
C       LETTERS, AND ASSOCIATED POINTERS IN THE FORMAT THAT WILL BE USED
C       IN THE SUFFIX PROGRAM.
C             INPUT FILE IS READ IN BINARY (I.E. NO CHANGE IS MADE TO
C       FORMAT OF THE RECORD) FROM LOGICAL UNIT 20
C             THE OUTPUT FILE IS WRITTEN IN BINARY TO LOGICAL UNIT 30
C
      DIMENSION ISUFF1(18),ISUFF2(18),KSUFF1(3,150),KSUFF2(3,375),
     1  NOSUF1(150),INDSF2(150),ISFCN2(150),NOSUF2(375),IEXPIN(375),
     2  IEXPCT(375),ILETIN(375),ILETCT(375),IWORDX(3, 800),IWRDXL( 800),
     3  LETXCP( 50),KOLD1(18),KOLD2(18),IEXCEP(18)
      DATA NOCHAR,JWORDS,MAXSF1,MAXSF2,MAXEXP,MAXLET/18,3,150,375,
     1  800,50/
      DATA KE,KL,IBLANK,LABEL/1HE,1HL,6H      ,0/,LZ/0772020202020/
      DATA LIMPAG/60/
C
C
      DO 10000 K=1,NOCHAR
      KOLD1(K) = IBLANK
      KOLD2(K) = IBLANK
10000 CONTINUE
C       READ IN FIRST SUFFIX RECORD.
      READ (20)        ISUFF1,ISUFF2,ITYPE,IEXCEP
C       GO TO PROCESS RECORD.
      GO TO 30000
C
C       GET NEXT RECORD.
C
20000 READ (20)        ISUFF1,ISUFF2,ITYPE,IEXCEP
      IF(ISUFF1(1).EQ.LZ.AND.ISUFF1(2).EQ.LZ.AND.ISUFF1(3).EQ.LZ)
     1   GO TO 80000
C       IS THE FIRST SUFFIX THE SAME AS THE PREVIOUS RECORD.
      DO 20100 K=1,NOCHAR
      IF(ISUFF1(K).NE.KOLD1(K)) GO TO 30000
      IF(ISUFF1(K).EQ.IBLANK) GO TO 20200
20100 CONTINUE
C       THE FIRST SUFFIX WAS THE SAME AS THE PREVIOUS RECORD.  IS THE
C       SECOND SUFFIX DIFFERENT.
20200 DO 20300 K=1,NOCHAR
      IF(ISUFF2(K).NE.KOLD2(K)) GO TO 20500
      IF(ISUFF2(K).EQ.IBLANK) GO TO 20400
20300 CONTINUE
20400 IF(ITYPE.EQ.IBLANK) GO TO 91000
      IF(ITYPE.EQ.KE) GO TO 40000
      IF(ITYPE.EQ.KL) GO TO 50000
      GO TO 91000
C
C   NEW SECOND SUFFIX
C
20500 IF(ITYPE.NE.IBLANK) GO TO 93000
21000 IF(NSUFF2.GE.MAXSF2) GO TO 97000
      NSUFF2 = NSUFF2 + 1
C       MOVE SUFFIX TO SAVE AREA.
      DO 21100 K=1,NOCHAR
      KOLD2(K) = ISUFF2(K)
```

```
21100 CONTINUE
C * * * * *
C       COMPCT IS A SUBROUTINE WHICH PACKS CHARACTERS.  SEE USERS MANUAL
C * * * * *
      CALL COMPCT(KSUFF2(1,NSUFF2),ISUFF2(1),NOCHAR)
      K = 1
21200 IF(ISUFF2(K+1).EQ.IBLANK) GO TO 21300
      K = K + 1
      IF(K.LT.NOCHAR) GO TO 21200
21300 NOSUF2(NSUFF2) = K
      ISFCN2(NSUFF1) = ISFCN2(NSUFF1) + 1
      GO TO 20000
C
C   NEW FIRST SUFFIX
C
30000 IF(ITYPE.NE.IBLANK) GO TO 93000
      IF(NSUFF1.GE.MAXSF1) GO TO 97500
      NSUFF1 = NSUFF1 + 1
      IF(NSUFF2.GE.MAXSF2) GO TO 97000
      NSUFF2 = NSUFF2 + 1
      DO 30100 K=1,NOCHAR
      KOLD1(K) = ISUFF1(K)
      KOLD2(K) = ISUFF2(K)
30100 CONTINUE
C * * * * *
C       COMPCT IS A SUBROUTINE WHICH PACKS CHARACTERS.  SEE USERS MANUAL
C * * * * *
      CALL COMPCT(KSUFF1(1,NSUFF1),ISUFF1(1),NOCHAR)
      CALL COMPCT(KSUFF2(1,NSUFF2),ISUFF2(1),NOCHAR)
      K = 1
30200 IF(ISUFF1(K+1).EQ.IBLANK) GO TO 30300
      K = K + 1
      IF(K.LT.NOCHAR) GO TO 30200
30300 NOSUF1(NSUFF1) = K
      K = 1
30400 IF(ISUFF2(K+1).EQ.IBLANK) GO TO 30500
      K = K + 1
      IF(K.LT.NOCHAR) GO TO 30400
30500 NOSUF2(NSUFF2) = K
      INDSF2(NSUFF1) = NSUFF2
      ISFCN2(NSUFF1) = 1
      GO TO 20000
C
C   NEW EXCEPTION WORD
C
40000 IF(NEXCEP.GE.MAXEXP) GO TO 98000
      NEXCEP = NEXCEP + 1
      IF(IEXPIN(NSUFF2).NE.0) GO TO 40100
      IEXPIN(NSUFF2) = NEXCEP
      IEXPCT(NSUFF2) = 1
      GO TO 40150
40100 IEXPCT(NSUFF2) = IEXPCT(NSUFF2) + 1
C * * * * *
C       COMPCT IS A SUBROUTINE WHICH PACKS CHARACTERS.  SEE USERS MANUAL
C * * * * *
40150 CALL COMPCT(IWORDX(1,NEXCEP),IEXCEP(1),NOCHAR)
      K = 1
40200 IF(IEXCEP(K+1).EQ.IBLANK) GO TO 40300
      K = K + 1
      IF(K.LT.NOCHAR) GO TO 40200
```

```
40300 IWRDXL(NEXCEP) = K
      GO TO 20000
C
C   NEW EXCEPTION LETTER
C
50000 IF(NLETTR.GE.MAXLET) GO TO 98500
      NLETTR = NLETTR + 1
      IF(ILETIN(NSUFF2).NE.0) GO TO 50100
      ILETIN(NSUFF2) = NLETTR
      ILETCT(NSUFF2) = 1
      GO TO 50200
50100 ILETCT(NSUFF2) = ILETCT(NSUFF2) + 1
50200 LETXCP(NLETTR) = IEXCEP(1)
      GO TO 20000
C
C   OUTPUT SECTION
C
C       WRITE TOTAL NUMBER OV SUFFIX(S AND EXCEPTION WORDS AND LETTERS.
80000 WRITE (30)       NSUFF1,NSUFF2,NEXCEP,NLETTR
C       WRITE FIRST SUFFIX TABLE.
      WRITE (30)        ((KSUFF1(I,J),I=1,JWORDS),J=1,NSUFF1)
C       WRITE POINTER TABLES ASSOCIATED WITH FIRST SUFFIX.
      WRITE (30)        (NOSUF1(J),INDSF2(J),ISFCN2(J),J=1,NSUFF1)
C       WRITE SECOND SUFFIX TABLE
      WRITE (30)        ((KSUFF2(I,J),I=1,JWORDS),J=1,NSUFF2)
C       WRITE TABLES OF POINTERS ASSOCIATED WITH SECOND SUFFIX.
      WRITE (30)        (NOSUF2(J),IEXPIN(J),IEXPCT(J),ILETIN(J),
     1  ILETCT(J),J=1,NSUFF2)
C       WRITE TABLES FO EXCEPTION WORDS AND LETTERS.
      WRITE (30)        ((IWORDX(I,J),I=1,JWORDS),J=1,NEXCEP)
      WRITE (30)        (IWRDXL(J),J=1,NEXCEP)
      WRITE (30)       (LETXCP(J),J=1,NLETTR)
      WRITE (6,80100) NSUFF1,NSUFF2,NEXCEP,NLETTR
80100 FORMAT (1H1,10H NSUFF1 = ,I5/10H NSUFF2 = ,I5/10H NEXCEP = ,I5/10H
     1 NLETTR = ,I5)
      STOP
C
C       ERROR MESSAGES.
C
91000 WRITE (6,91100) ITYPE
91100 FORMAT (1X,28H* * * * ILLEGAL TYPE FIELD, ,A1,37H, FOR UPDATE RECO
     1RD.  JOB TERMINATED.)
      WRITE (6,91200) ISUFF1,ISUFF2,ITYPE,IEXCEP
91200 FORMAT (2(1X,18A1),2X,A1,1X,18A1)
      STOP
93000 WRITE (6,93100)
93100 FORMAT (1X,69H* * * * ILLEGAL TYPE FIELD IN RECORD FOR NEW SUFFIX.
     1  JOB TERMINATED.)
      WRITE (6,91200) ISUFF1,ISUFF2,ITYPE,IEXCEP
      STOP
97000 WRITE (6,97100) MAXSF2
97100 FORMAT (1X,17H* * * * LIMIT OF ,I5,44H SUFFIX 2 RECORDS EXCEEDED.
     1 JOB TERMINATED.)
      GO TO 80000
97500 WRITE (6,97510) MAXSF1
97510 FORMAT (1X,17H* * * * LIMIT OF ,I5,44H SUFFIX 1 RECORDS EXCEEDED.
     1 JOB TERMINATED.)
      GO TO 80000
98000 WRITE (6,98100) MAXEXP
98100 FORMAT (1X,17H* * * * LIMIT OF ,I5,50H EXCEPTION WORD RECORDS EXCE
```

```
     1EDED.  JOB TERMINATED.)
      GO TO 80000
98500 WRITE (6,98510) MAXLET
98510 FORMAT (1X,17H* * * * LIMIT OF ,I5,52H EXCEPTION LETTER RECORDS EX
     1CEEDED.  JOB TERMINATED.)
      GO TO 80000
      END
```

```
CSUFFX1         SUFFIX LOGIC MODULE
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      COMMON /INIT2/ IPRINT,NPAGE,NEWPAG
      COMMON /IMAT/ IRWORD(3,500),ICOUNT(500),IRMATC(500),IRTSAV(3)
      COMMON /READ/ INPRFL,INWRDL,INPFIX(8),INWORD(18),NCOUNT,NEWORD(18)
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,JFNFLG,JPNFLG,
     1  ICNTNT
      COMMON /RPRT/ ITITLE(14),IRPTHD(39),IRPTDT(39),JRPTDT(39),
     1  ITOTFM(13),IRPTTL(13)
      DATA IBLANK,LS,LAPOST,LE/1H ,1HS,1H',1HE/,IWDEOF/0/
      DATA NBYTES,NUMBRT,NBASE,JWORDS/6,500,15,3/
      DATA IFIRST,NO/0,1/
C
      NOCHAR = 18
      NCHARP = 8
      NSTEM = NOCHAR - 3
C       READ IN INITIAL TABLES AND CONTROL INFORMATION
      CALL SFINIT
      CALL SFRDIN
      CALL SFMTC1
      IF(IWDEOF.EQ.1) CALL SFINAL(NEWPAG,IPRINT)
      ISAVEC = 1
      DO 12000 I=1,3
      IRTSAV(I) = NEWORD(I)
12000 CONTINUE
C
C         GET NEXT TEXT WORD
C
13000 CALL SFREAD(IWDEOF)
C
C         REMOVE PLURAL AND POSSESSIVE FORMS.
C            THIS ROUTINE REMOVES 'S, S', S OR S, (IF WORD DOES NOT END IN
C            SS), OR ES OR ES, (IF ES IS PRECEDED BY S AS IN GUESSES)
C            FROM THE END OF THE TEXT WORD FOR PURPOSES OF ROOT GROUPING.
C            THE ACTUAL TEXT RECORDS, HOWEVER, ARE NOT ALTERED.
C
      J = INWRDL - 1
      IF(INWORD(INWRDL).NE.LS) GO TO 13200
      IF(INWORD(J).NE.LAPOST) GO TO 13100
C
C         WORD ENDS IN 'S
C
      INWORD(INWRDL) = IBLANK
      INWORD(J) = IBLANK
      INWRDL = INWRDL - 2
      GO TO 13500
C
13100 IF(INWORD(J).EQ.LS) GO TO 13500
C         WORD ENDS IN S OR ES
C
      INWORD(INWRDL) = IBLANK
      INWRDL = J
      IF(INWORD(INWRDL).NE.LE) GO TO 13500
      J = INWRDL - 1
      IF(INWORD(J).NE.LS) GO TO 13500
      INWORD(INWRDL) = IBLANK
      INWRDL = J
      GO TO 13500
13200 IF(INWORD(INWRDL).NE.LAPOST) GO TO 13500
```

```
C
C        WORD ENDS IN ,
C
      INWORD(INWRDL) = IBLANK
      INWRDL = J
      J = J - 1
      IF(INWORD(INWRDL).EQ.LS) GO TO 13100
13500 IF(ISAVEC.GT.NUMBRT) GO TO 16000
13700 NTEMP = INWRDL - 3
      IF(NTEMP.GE.0) GO TO 14000
      ICOUNT(ISAVEC) = NCOUNT
      GO TO 14500
14000 CALL COMPCT(IRWORD(1,ISAVEC),INWORD(4),NSTEM)
      ICOUNT(ISAVEC) = NCOUNT
      IF(IWDEOF.EQ.1) GO TO 14500
C
C        TEST TO SEE IF THE NEW WORD STARTS A NEW ROOT GROUPING SET
C
14050 DO 14100 I=1,3
      IF(IRTSAV(I).NE.NEWORD(I)) GO TO 14500
14100 CONTINUE
14200 ISAVEC = ISAVEC + 1
      GO TO 13000
C
C        CALCULATE MATCH COUNTS
C
14500 CALL SFMATC
C
C        COMPUTE FREQUENCIES AND OUTPUT
C
      CALL SFOUT1(NPAGE,IPRINT)
C
C        CHECK TO SEE IF END-OF-FILE ON INPUT.  IF SO, GO TO SFINAL.
C
      IF(IWDEOF.EQ.1) CALL SFINAL(NEWPAG,IPRINT)
C
C        INITIALIZE FOR NEW SET
C
14550 DO 14600 I=1,3
      IRTSAV(I) = NEWORD(I)
14600 CONTINUE
      ISAVEC = 1
      GO TO 13000
C
16000 WRITE  (6,16100)
16100 FORMAT (1X,61H* * * * TOO LITTLE ROOM IN TEMPORARY TABLES.  JOB TE
     1RMINATED.)
      STOP
      END
```

```
CSFOUT1         MODULE TO LIST OUT WORDS AND MATCH COUNTS
C
C                SUBROUTINE SFOUT1 EMPTIES OUT TABLE IRMATC AFTER THE
C          MATCH COUNTS HAVE BEEN ASSIGNED BY SUBROUTINE CSFMATC
C
       SUBROUTINE SFOUT1(NPAGE,IPRINT)
       COMMON /ALL/ NOCHAR,NCHARP,NSTEM
       COMMON /RPRT/ ITITLE(14),IRPTHD(39),IRPTDT(39),JRPTDT(39),
      1  ITOT(13),IRPTTL(13)
       COMMON /IMAT/ IRWORD(3,500),ICOUNT(500),IRMATC(500),IRTSAV(3)
       COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,CDUMMY(2),ICNTNT
       DIMENSION MPOINT(500),MCFREQ(500),IOPFIX(8),IOWORD(18),INDEXO(3,8)
       DATA LINES/200/,IBLANK/6H      /,IDATE/6H      /,MPAGE/1/
40000  MCTEST = 0
       M = 0
C
C         ACCUMULATE MATCH COUNT TOTALS
C          THE TABLE ,IRMATC, CONTAINS THE MATCHCOUNT ASSIGNED TO EACH
C            WORD TYPE BY ROUTINE SFMATC.
C          TABLE ,ICOUNT, CONTAINS THE NUMBER OF TIMES EACH TYPE OCCURED
C            IN THE TEXT
C          TABLE ,MCFREQ, IS USED TO ACCUMULATE THE TOTAL FREQUENCY FOR
C            EACH MATCH COUNT
C          TABLE ,MPOINT, LINKS EACH TYPE TO ITS MATCH COUMT TOTAL IN
C            ,MCFREQ,
C
       DO 4035 I=1,ISAVEC
       IF(IRMATC(I).LE.MCTEST) GO TO 4035
       M = M + 1
       MPOINT(I) = M
       MCTEST = IRMATC(I)
       MCFREQ(M) = ICOUNT(I)
       IF(I.GE.ISAVEC) GO TO 4035
       IP1 = I + 1
       DO 4030 J=IP1,ISAVEC
       IF(IRMATC(J).NE.MCTEST) GO TO 4030
       MCFREQ(M) = MCFREQ(M) + ICOUNT(J)
       MPOINT(J) = M
 4030  CONTINUE
 4035  CONTINUE
C
C          TOTALS HAVE BEEN CALCULATED.  DO OUTPUT ROUTINE.
C
       IF(IPRINT.EQ.0) GO TO 4100
C
C          WRITE OUT PRINTED REPORT
C
       IF(LINES.LT.(NPAGE-6)) GO TO 4040
       NEWPAG = 1
       GO TO 41000
 4040  LINES = LINES + 2
       WRITE  (6,4050)
 4050  FORMAT (1HO)
       K = 1
       MATOUT = MCTEST - M
C
C         WRITE OUT SUMMARY RECORDS FOR MATCH COUNTS
C
 4060  MATOUT = MATOUT + 1
```

```
      WRITE  (6,ITOT) MCFREQ(K),MATOUT
      LINES = LINES + 1
      K = K + 1
      IF(K.GT.M) GO TO  4080
      IF(LINES.LT.NPAGE) GO TO 4060
      NEWPAG = 2
      GO TO 41000
 4080 IF(LINES.LT.(NPAGE-2)) GO TO 4085
      NEWPAG = 5
      GO TO 41000
 4085 WRITE  (6,4090)
C * * * * *   FORMAT   * * * * *
 4090 FORMAT (1X)
C * * * * *
      LINES = LINES + 1
C
C       WRITE OUT CONTENT WORD TOKEN RECORDS
C
 4100 REWIND 14
      I = 1
 4110 IMAT = MPOINT(I)
      KCOUNT = ICOUNT(I)
      ICNTNT = ICNTNT + KCOUNT
      LTEMP = KCOUNT/3
      LRMAIN = KCOUNT - (3 * LTEMP)
      IF(LTEMP.GT.0) GO TO 4130
      J1 = KCOUNT
      GO TO 4140
 4130 J1 = 3
C
C         WORD TYPE RECORDS AND INDEXES ARE ON FILE 14.  OUTPUT FOR
C           THESAUR RUNS GOES ON FILE 13
C
 4140 READ (14)  IOPRFL,IOWRDL,IOPFIX,IOWORD
      DO 4180 K=1,J1
      READ (14)       (INDEXO(K,K1),K1=1,8)
      WRITE (13) (INDEXO(K,K1),K1=1,8),IRMATC(I),MCFREQ(IMAT),KCOUNT,
     1   IOPRFL,IOWRDL,IOPFIX,IOWORD
 4180 CONTINUE
      IF(IPRINT.EQ.0) GO TO 4200
C
C         WRITE DETAIL LINE 1.
C
      WRITE  (6,IRPTDT) ICOUNT(I),IRMATC(I),IOPFIX,IOWORD,
     1  ((INDEXO(K1,K2),K2=2,6),K1=1,J1)
      LINES = LINES + 1
      IF(LINES.LT.NPAGE) GO TO 4200
      NEWPAG = 6
      GO TO 41000
 4200 IF(LTEMP.EQ.0) GO TO 300
C
C         THERE WERE MORE THAN 3 OCCURENCES OF THE WORD TYPE IN THE TEXT
C
      LTEMP = LTEMP - 1
      IF(LTEMP.EQ.0) GO TO 4250
      J2 = 1
C
C         READ INDEX ENTRIES 3 AT A TIME
C
 4205 DO 4210 K=1,3
```

```
      READ (14)       (INDEXO(K,K1),K1=1,8)
      WRITE (13) (INDEXO(K,K1),K1=1,8),IRMATC(I),MCFREQ(IMAT),KCOUNT,
     1   IOPRFL,IOWRDL,IOPFIX,IOWORD
 4210 CONTINUE
      IF(IPRINT.EQ.0) GO TO 4240
      IF(LINES.LT.NPAGE) GO TO 4220
      NEWPAG = 3
      GO TO 41000
C
C        WRITE DETAIL LINE 2
C
 4220 WRITE  (6,JRPTDT) ((INDEXO(K1,K2),K2=2,6),K1=1,3)
      LINES = LINES + 1
 4240 J2 = J2 + 1
      IF(J2.LE.LTEMP) GO TO 4205
 4250 IF(LRMAIN.EQ.0) GO TO 300
C
C        LESS THAN THREE INDEX ENTRIES ARE LEFT.  WRITE THEM OUT
C
      DO 4260 K=1,LRMAIN
      READ (14)       (INDEXO(K,K1),K1=1,8)
      WRITE (13) (INDEXO(K,K1),K1=1,8),IRMATC(I),MCFREQ(IMAT),KCOUNT,
     1   IOPRFL,IOWRDL,IOPFIX,IOWORD
 4260 CONTINUE
      IF(IPRINT.EQ.0) GO TO 300
      IF(LINES.LT.NPAGE) GO TO 4270
      NEWPAG = 4
      GO TO 41000
 4270 WRITE  (6,JRPTDT) ((INDEXO(K1,K2),K2=2,6),K1=1,LRMAIN)
  300 I = I + 1
      IF(I.LE.ISAVEC) GO TO 4110
C
C        REWIND FILE 14 FOR NEW GROUP OF WORDS.
C
40400 REWIND 14
      RETURN
C
C
C        NEW PAGE ROUTINE
C
41000 WRITE  (6,IRPTHD) IRPTTL,MPAGE
      MPAGE = MPAGE + 1
      LINES = 7
      GO TO (4040,4060,4220,4270,4100,4200), NEWPAG
      END
```

```
CSFINIT        SFINIT-MODULE TO INITIALIZE DATA AREAS
C
C              SUBROUTINE SFINIT READS IN CONTROL PARAMETERS AND
C         INITIALIZES DATA TABLES.
C
      SUBROUTINE SFINIT
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,JFNFLG,JPNFLG,
     1  ICNTNT
      COMMON /INIT2/ IPRINT,NPAGE,NEWPAG
      COMMON /RPRT/ ITITLE(14),IRPTHD(39),IRPTDT(39),JRPTDT(39),
     1  ITOTFM(13),IRPTTL(13)
      DIMENSION IOUT(2),KPR(9),NUM(3)
      INTEGER CONTRL(80)
      DATA LA,LC,LE,LF,LH,LL,LN,LP,LS,LV,LY/1HA,1HC,1HE,1HF,1HH,1HL,
     1  1HN,1HP,1HS,1HV,1HY/
      DATA (IOUT(I),I=1,2),(NUM(I),I=1,3)/3HYES,2HNO,1H1,1H2,1H3/
      DATA LEQ,LAPOST,LCOM,IBLANK/1H=,1H',1H,,1H /
      DATA MLEV,MPN,MFN/3*0/
C
      IPRINT = 1
      NPAGE = 60
      NEWPAG = 60
      LEVELP = LA
      JFNFLG = 1
      JPNFLG = 1
      READ (5,25100) CONTRL
25100 FORMAT (80A1)
      I = 1
      IF(CONTRL(I).EQ.IBLANK) GO TO 27000
C
C        IS THE PARAMETER THE FUNCT PARAMETER
C
25200 IF(CONTRL(I).EQ.LF) GO TO 26000
C
C        IS THE PARAMETER THE PUNCT PARAMETER
C
25300 IF(CONTRL(I).EQ.LP) GO TO 26500
C
C        IF THE PARAMETER IS NOT THE LEVEL PARAMETER, THEN ERROR
C
      GO TO 29500
C
C
C        PROCESS FUNCTION PARAMETER
C
26000 I = I + 5
      IF(CONTRL(I).NE.LEQ) GO TO 29500
      I = I + 1
      IF(CONTRL(I).NE.LY) GO TO 26100
      I = I + 3
      MFN = 1
      GO TO 26200
26100 IF(CONTRL(I).NE.LN) GO TO 29500
      JFNFLG = 0
      I = I + 2
      MFN = 2
26200 IF(CONTRL(I).EQ.IBLANK) GO TO 27000
      IF(CONTRL(I).NE.LCOM) GO TO 29500
```

```
      I = I + 1
      GO TO 25300
C
C        PROCESS PUNCTUATION PARAMETER
C
26500 I = I + 5
      IF(CONTRL(I).NE.LEQ) GO TO 29500
      I = I + 1
      IF(CONTRL(I).NE.LY) GO TO 26600
      I = I + 3
      MPN = 1
      GO TO 26700
26600 IF(CONTRL(I).NE.LN) GO TO 29500
      JPNFLG = 0
      I = I + 2
      MPN = 2
26700 IF(CONTRL(I).EQ.IBLANK) GO TO 27000
      IF(CONTRL(I).NE.LCOM) GO TO 29500
      I = I + 1
      GO TO 25200
C
C        WRAP UP
C
27000 WRITE (6,28200)
      IF(MFN.EQ.0) GO TO 27100
      WRITE (6,27600) IOUT(MFN)
      GO TO 27200
27100 WRITE (6,27700)
27200 IF(MPN.EQ.0) GO TO 27300
      WRITE (6,27800) IOUT(MPN)
      GO TO 10000
27300 WRITE (6,27900)
C
C        FORMATS
C
27600 FORMAT (1X,13HFUNCT SET TO ,1A3)
27700 FORMAT (1X,27HFUNCT SET TO YES BY DEFAULT)
27800 FORMAT (1X,13HPUNCT SET TO ,1A3)
27900 FORMAT (1X,27HPUNCT SET TO YES BY DEFAULT)
28200 FORMAT (1H1,20X,24HPARAMETER SPECIFICATIONS//)
C
10000 READ (5,10100) KPR,KPG,KPAG
10100 FORMAT (9A1,2I5)
      I = 1
      IF(KPR(I).EQ.IBLANK) GO TO 10300
      IF(KPR(I).NE.LP) GO TO 18000
      I = 6
      IF(KPR(I).NE.LEQ) GO TO 18000
      I = I + 1
      IF(KPR(I).EQ.LN) GO TO 10200
      IF(KPR(I).NE.LY) GO TO 18000
C
C        PRINT = YES
C
      WRITE (6,10401) IOUT(1)
      GO TO 10500
C
C        PRINT=NO
C
10200 IPRINT = 0
```

```
      WRITE (6,10401) IOUT(2)
      GO TO 10900
C
C        PRINT PARAMETER NOT SPECIFIED
C
10300 WRITE (6,10402)
      GO TO 10500
C
C        PRINT PARAMETER MESSAGE FORMATS
C
10401 FORMAT (1X,13HPRINT SET TO ,A3)
10402 FORMAT (1X,27HPRINT SET TO YES BY DEFAULT)
C
C        PROCESS OUTPUT PAGE LENGTH PARAMETER
C
10500 IF(KPG.EQ.0) GO TO 10600
      IF(KPG.GT.150) GO TO 10700
C
C        PARAMETER IS LEGAL
C
      NPAGE = KPG
      WRITE (6,10801) NPAGE
      GO TO 10900
C
C        PAGE LENGTH PARAMETER NOT SPECIFIED
C
10600 WRITE (6,10802) NPAGE
      GO TO 10900
C
C        ILLEGAL SIZE
C
10700 WRITE (6,10803)
      STOP
C
C        PAGE LENGTH PARAMETER MESSAGES
C
10801 FORMAT (1X,22HOUTPUT PAGE LENGTH IS ,I5)
10802 FORMAT (1X,22HOUTPUT PAGE LENGTH IS ,I5,11H BY DEFAULT)
10803 FORMAT (1X,56HOUTPUT PAGE LENGTH IS GREATER THAN 150.  JOB TERMINA
     1TED.)
C
C        PROCESS FUNCT/PUNCT REPORT PAGE LENGTH PARAMETER
C
10900 IF(KPAG.EQ.0) GO TO 11000
      IF(KPAG.GT.150) GO TO 11100
C        PARAMETER IS LEGAL
      NEWPAG = KPAG
      WRITE (6,11201) NEWPAG
      GO TO 11300
C
C        FUNCT/PUNCT REPORT PAGE LENGTH NOT SPECIFIED
C
11000 WRITE (6,11202) NEWPAG
      GO TO 11300
C
C        ILLEGAL SIZE
C
11100 WRITE (6,11203)
      STOP
C
```

```
C         FUNCT/PUNCT REPORT PAGE LENGTH MESSAGES
C
11201 FORMAT (1X,28HFUNCT REPORT PAGE LENGTH IS ,I5)
11202 FORMAT (1X,28HFUNCT REPORT PAGE LENGTH IS ,I5,11H BY DEFAULT)
11203 FORMAT (1X,62HFUNCT REPORT PAGE LENGTH IS GREATER THAN 150.  JOB T
     1ERMINATED )
C
C         READ FORMAT CONTROL PARAMETERS
C
11300 READ (5,11400) CONTRL
11400 FORMAT (80A1)
      I = 1
      IF(CONTRL(I).NE.LE) GO TO 18500
      I = I + 5
      IF(CONTRL(I).NE.LEQ) GO TO 18500
      I = I + 1
      IF(CONTRL(I).EQ.LN) GO TO 11500
C
C         TITLE FOR FUNCT/PUNCT REPORT WILL BE PRESENT
C
      KERR = 1
      I = I + 4
      GO TO 11600
11500 IF(CONTRL(I).NE.LN) GO TO 18500
      KERR = 0
      I = I + 3
11600 IF(CONTRL(I).NE.LH) GO TO 18500
      I = I + 7
      IF(CONTRL(I).NE.LEQ) GO TO 18500
      I = I + 1
C
C         FIND NUMBER OF HEADING FORMAT CARDS
C
      DO 11700 J=1,3
      IF(CONTRL(I).EQ.NUM(J)) GO TO 11800
11700 CONTINUE
      GO TO 18500
11800 INH = J * 13
      IF(CONTRL(I+1).NE.LCOM) GO TO 18500
      I = I + 8
      IF(CONTRL(I).NE.NUM(1)) GO TO 18500
      I = I + 2
C
C         FIND NUMBER OF FORMAT CARDS FOR DETAIL LINE 1
C
      DO 11900 J=1,3
      IF(CONTRL(I).EQ.NUM(J)) GO TO 12000
11900 CONTINUE
      GO TO 18500
12000 IN1 = J * 13
      IF(CONTRL(I+1).NE.LCOM) GO TO 18500
      I = I + 8
      IF(CONTRL(I).NE.NUM(2)) GO TO 18500
      I = I + 2
C
C         FIND NUMBER OF FORMAT CARDS FOR DETAIL LINE 2
C
      DO 12100 J = 1,3
      IF(CONTRL(I).EQ.NUM(J)) GO TO 12200
12100 CONTINUE
```

```
      GO TO 18500
12200 IN2 = J * 13
      IF(CONTRL(I+1).NE.IBLANK) GO TO 18500
      IF(KERR.EQ.0) GO TO 12400
C
C        READ IN FUNCTION AND PUNCTUATION LISTING TITLE
C
      READ (5,12300) ITITLE
12300 FORMAT (13A6,A2)
C
C        READ IN FORMAT CARDS
C
12400 READ (5,12500) (IRPTHD(I),I=1,INH)
      READ (5,12500) (IRPTDT(I),I=1,IN1)
      READ (5,12500) (JRPTDT(I),I=1,IN2)
      READ (5,12500) ITOTFM
C        READ IN REPORT TITLE
      READ (5,12500) IRPTTL
12500 FORMAT (13A6)
C
      RETURN
C
C
18000 WRITE(6,18600) I,KPR
18600 FORMAT (1X,29HCONTROL CARD ERROR IN COLUMN ,I2,18H.  JOB TERMINATE
     1D./1X 6HKPR = ,9A1)
      STOP
185000 CONTINUE
29500 WRITE (6,29600) I,CONTRL
29600 FORMAT(1X,29HCONTROL CARD ERROR IN COLUMN ,I2,18H.  JOB TERMINATED
     1./1X,10HCARD WAS--,80A1)
      STOP
      END
```

```
CSFREAD         SFREAD--MODULE TO READ IN TEXT RECORDS
C
C               SUBROUTINE SFREAD OBTAINS SUCCESSIVE INDEXED RECORDS,
C         PRE-PROCESSES FUNCTION WORDS AND PUNCTUATION (SEE SUBROUTINE
C         SFINAL FOR PRINTING OF FUNCTION WORDS AND PUNCTUATION) AND
C         RETURNS OTHER WORDS TO THE MAIN ROUTINE, SUFFX1.
C
      SUBROUTINE SFREAD(IPASS)
      COMMON /READ/ INPRFL,INWRDL,INPFIX(8),INWORD(18),NCOUNT,NEWORD(18)
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,JFNFLG,JPNFLG,
     1  ICNTNT
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      DIMENSION NEWPRE(8),INDEXN(8)
      DIMENSION IFNCWD(18)
      EQUIVALENCE (INDEXN(2),INDVOL),(INDEXN(3),INDCHP)
      EQUIVALENCE (INDEXN(4),INDPAR),(INDEXN(5),INDSEN)
      DATA LA,LV,LC,IBLANK,LP/1HA,1HV,1HC,1H ,1HP/
      DATA JMATOT/0/
C
C         MOVE WORD AND PREFIX
C
19000 DO 19050 KL=1,NCHARP
      INPFIX(KL) = NEWPRE(KL)
19050 CONTINUE
      DO 19100 KL=1,NOCHAR
      INWORD(KL) = NEWORD(KL)
19100 CONTINUE
      INPRFL = NEWPFL
      INWRDL = NEWWDL
      NCOUNT = 0
C
C         WRITE TO TEMPORARY FILE.  WILL BE WRITTEN OUT IN SFOUT1.
C
      WRITE (14) INPRFL,INWRDL,INPFIX,INWORD
19150 WRITE (14) INDEXN
      NCOUNT = NCOUNT + 1
C
C         READ NEXT TEXT WORD
C
      READ (15) INDEXN,NEWPFL,NEWWDL,NEWPRE,NEWORD
      IF(INDEXN(1).NE.0) GO TO 19200
C
C         END OF FILE
      IPASS = 1
      RETURN
C
C         IS THIS WORD THE SAME AS THE PREVIOUS WORD
C
19200 IF(NEWPFL.NE.INPRFL) GO TO 19400
      IF(INPRFL.EQ.0) GO TO 19275
      DO 19250 KL=1,INPRFL
      IF(NEWPRE(KL).NE.INPFIX(KL)) GO TO 19400
19250 CONTINUE
19275 IF(NEWWDL.NE.INWRDL) GO TO 19400
19300 DO 19350 KL=1,INWRDL
      IF(NEWORD(KL).NE.INWORD(KL)) GO TO 19400
19350 CONTINUE
      GO TO 19150
C
```

```
C         WORD WAS DIFFERENT
C
19400 INCNT = INCNT + NCOUNT
20000 IF(IFNTYP.EQ.0) RETURN
C
C 20100* IS TEXT WORD EQUAL TO SUFUN FILE WORD
C
20100 IF(NEWPRE(1).NE.IBLANK) RETURN
      IF(NEWWDL.LT.NOCHAR) NEWORD(NEWWDL+1) = IBLANK
      IF(IFUNCL.LT.NOCHAR) IFNCWD(IFUNCL+1) = IBLANK
20200 DO 20300 I=1,NOCHAR
C  LCOMP IS A SUBROUTINE THAT COMPARES TWO COMPUTER WORDS LOGICALLY
C    (ON COLLATING SEQUENCE)
      ITEST = LCOMP(NEWORD(I),IFNCWD(I))
      IF(ITEST) 21600,20250,21100
20250 IF(NEWORD(I).EQ.IBLANK) GO TO 20350
20300 CONTINUE
C
C  WORDS ARE EQUAL. IS SUFUN FILE WORD A FUNCTION WORD OR PUNCTUATION
C
20350 IF(IFNTYP.EQ.1) GO TO 20400
C
C  PROCESS FUNCTION WORD
      IF(JFNFLG.EQ.0) GO TO 20500
      JMATOT = 99998
      GO TO 20500
C
C  20400*  PROCESS PUNCTUATION RECORD
C
20400 IF(JPNFLG.EQ.0) GO TO 20500
      JMATOT = 99999
C
C  20500*  WRITE TEMPORARY FILE.  FILE WILL BE WRITTEN OUT IN
C        SUBROUTINE SFINAL.
C
20500 IF(JMATOT.EQ.0) GO TO 20700
      WRITE (18) JMATOT,NEWPFL,NEWWDL,NEWPRE,NEWORD
20600 WRITE (18) INDEXN
      KCURR = KCURR + 1
20700 READ (15) INDEXN,NEWPFL,NEWWDL,NEWPRE,NEWORD
      IF(INDEXN(1).NE.0) GO TO 20750
      IPASS = 1
      IF(JMATOT.EQ.0) RETURN
      GO TO 21000
20750 IF(NEWPRE(1).NE.IBLANK) GO TO 20900
      IF(NEWWDL.NE.IFUNCL) GO TO 20900
      DO 20800 I=1,NEWWDL
      IF(NEWORD(I).NE.IFNCWD(I)) GO TO 20900
20800 CONTINUE
      IF(JMATOT.EQ.0) GO TO 20700
      GO TO 20600
20900 IF(JMATOT.EQ.0) GO TO 21100
21000 WRITE (17) KCURR,IFNTYP
      INCNT = INCNT + KCURR
      IF(IFNTYP.EQ.1) KPUNCT=KPUNCT+KCURR
      IF(IFNTYP.EQ.2) KFUNCT=KFUNCT+KCURR
      KCURR = 0
      IF(IPASS.EQ.1) RETURN
21100 READ (35) IFNTYP,IFUNCL,IFNCWD
      IF(IFNTYP.EQ.0) RETURN
```

```
      GO TO 20100
C
21600 RETURN
C
C
      ENTRY SFRDIN
      REWIND 35
      READ (35) IFNTYP,IFUNCL,IFNCWD
      READ (15) INDEXN,NEWPFL,NEWWDL,NEWPRE,NEWORD
      GO TO 20100
      END
```

```
CSFMATC        SFMATC--MODULE TO CALCULATE MATCH COUNTS
      SUBROUTINE SFMATC
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,CDUMMY(3)
      COMMON /IMAT/ IRWORD(3,500),ICOUNT(500),IRMATC(500),IRTSAV(3)
      DIMENSION ISUFF1(3,150),NOSUF1(150),INDSF2(150),ISFCN2(150),
     1  ISUFF2(3,375),NOSUF2(375),IEXPIN(375),IEXCPT(18),IEXPCT(375),
     2  IWRDXL( 800),IWORDX(3, 800),ILETIN(375),ILETCT(375),LETXCP( 50),
     3  IWRDCK(18),KWORDI(18),KWORDJ(18),MSUFF1(15),MSUFF2(15),
     4  ISUFTM(15),NTEMP(15)
      EQUIVALENCE (ILOW1,ILOW2,ILOW3),(IHIGH1,IHIGH2,IHIGH3),
     1  (ICHCK1,ICHCK2,ICHCK3),(NOSUFC,LENEXP)
      DATA IBLANK/6H      /,JWORDS/3/,MATNUM/1/
C  CALCULATE-MATCH-COUNT-30000 SECTION.  CHECKS WORDS THAT HAVE BEEN
C    PLACED IN THE ROOT CHECK TABLE FOR COMMON ROOTS OR STEMS.  TOKENS
C    WITH MATCHING STEMS ARE ASSIGNED IDENTICAL MATCH COUNTS.
C
C
      IF(ISAVEC.NE.1) GO TO 10400
      IRMATC(1) = MATNUM
      MATNUM = MATNUM + 1
      RETURN
10400 DO 15 I=1,ISAVEC
   15 IRMATC(I) = 0
      ITEMP = ISAVEC
30000 DO 35000 I=1,ISAVEC
C  I IS THE INDEX FOR WORD 1.
C  LOOK FOR WORD1 (I) THAT HAS NOT BEEN ASSIGNED A MATCH COUNT
C
      IF(IRMATC(I).NE.0) GO TO 35000
C
C  ASSIGN MATCH COUNT TO WORD1.
C
      IRMATC(I) = MATNUM
C
C  UNPACK WORD1.  (I.E. CONVERT TO A1 FORMAT)
C
      IF(I.LT.ISAVEC) GO TO 30020
      MATNUM = MATNUM + 1
      GO TO 35000
30020 CALL BRKUP(KWORDI,IRWORD(1,I),3)
      ITEMP = ITEMP - 1
      JTEMP = ISAVEC + 1
      DO 34000 J = 1,ITEMP
C
C  START AT BOTTOM OF TABLE AND LOOK FOR A WORD THAT HAS NOT BEEN
C   ASSIGNED A MATCH COUNT.
C
      JTEMP = JTEMP - 1
      IF(IRMATC(JTEMP).NE.0) GO TO 34000
C
C  UNPACK WORDJ
C
      CALL BRKUP(KWORDJ,IRWORD(1,JTEMP),3)
      DO 30050 K=1,NSTEM
C
C  CHECK FOR POINT AT WHICH THE TWO WORDS DIFFER
C
      IF(KWORDI(K).NE.KWORDJ(K)) GO TO 30100
```

```
      IF(KWORDI(K).EQ.IBLANK) GO TO 30075
30050 CONTINUE
30075 IRMATC(JTEMP) = MATNUM
      GO TO 34000
C
C30100*  POINT OF DIFFERENCE HAS BEEN LOCATED.  THE REMAINDER OF BOTH
C    WORDS WILL BE TREATED AS POTENTIAL SUFFIXES
C
30100 KARG = (NSTEM + 1) - K
      MTRAN = K - 1
      DO 30200 M=1,KARG
      MTRAN = MTRAN + 1
      MSUFF1(M) = KWORDI(MTRAN)
      MSUFF2(M) = KWORDJ(MTRAN)
30200 CONTINUE
C  THE POTENTIAL SUFFIXES HAVE BEEN TRANSFERED TO WORK AREAS
C
      DO 30500 K1=1,KARG
C
C  THE SMALLER SUFFIX (BY COLLATING SEQUENCE) SHOULD BE IN HOLD AREA 1.
C    IF IT IS NOT, MAKE A TRANSFER.  LCOMP IS A SUBROUTINE THAT DOES A
C    LOGICAL COMPARE (I.E. BY COLLATING SEQUENCE) OF TWO COMPUTER
C    WORDS
C
      JTEST = LCOMP(MSUFF1(K1),MSUFF2(K1))
      IF(JTEST) 30700,30500,30300
30500 CONTINUE
C
C     SHOULD NEVER GET HERE
      GO TO 34500
C
C30300*  REMAINDER OF WORD1 IS HIGHER IN COLLATING SEQUENCE THAN
C    REMAINDER OF WORD 2
C
30300 DO 30400 N1=1,KARG
      NTEMP(N1) = MSUFF1(N1)
      MSUFF1(N1) = MSUFF2(N1)
      MSUFF2(N1) = NTEMP(N1)
30400 CONTINUE
C
C  NOW CHECK SUFFIX TABLES
      GO TO 30700
C
C30700*  DO BINARY SEARCH FOR SUFFIX 1
30700 ILOW1 = 1
      IHIGH1 = NSUFF1
30800 IF(ILOW1.GT.IHIGH1) GO TO 34000
      ICHCK1 = (ILOW1 + IHIGH1)/2
      NOSUFC = NOSUF1(ICHCK1)
      ICHAR = (NOSUFC + 5)/6
      CALL BRKUP(ISUFTM,ISUFF1(1,ICHCK1),ICHAR)
C
      DO 30900 L1=1,NOSUFC
      JTEST = LCOMP(MSUFF1(L1),ISUFTM(L1))
      IF(JTEST) 31100,30900,31200
30900 CONTINUE
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE SUFFIX IN
C    TABLE 1.  IF NO MORE CHARACTERS REMAIN IN THE POTENTIAL SUFFIX,
C    A MATCH HAS BEEN MADE.  OTHERWISE, THE POTENTIAL SUFFIX IS LARGER
```

```
C     THAN THE TABLE ENTRY
C
      IF(MSUFF1(NOSUFC+1).NE.IBLANK) GO TO 31200
      GO TO 31300
C
C31100*  THE POTENTIAL SUFFIX IS LESS THAN THE TABLE ENTRY
C
31100 IHIGH1 = ICHCK1 - 1
      GO TO 30800
C
C31200*  THE POTENTIAL SUFFIX IS GREATER THAN THE TABLE ENTRY
C
31200 ILOW1 = ICHCK1 + 1
      GO TO 30800
C
C
C31300*  SUFFIX 1 HAS BEEN MATCHED.  CHECK FOR SUFFIX 2.
C
31300 ILOW2 = INDSF2(ICHCK1)
      IHIGH2 = INDSF2(ICHCK1) + ISFCN2(ICHCK1) - 1
31400 IF(ILOW2.GT.IHIGH2) GO TO 34000
      ICHCK2 = (ILOW2 + IHIGH2)/2
      NOSUFC = NOSUF2(ICHCK2)
      ICHAR = (NOSUFC + 5)/6
      CALL BRKUP(ISUFTM,ISUFF2(1,ICHCK2),ICHAR)
      DO 31500 L2=1,NOSUFC
      JTEST = LCOMP(MSUFF2(L2),ISUFTM(L2))
      IF(JTEST) 31600,31500,31700
31500 CONTINUE
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE SUFFIX IN
C    TABLE 2.  IF NO MORE CHARACTERS REMAIN IN THE POTENTIAL SUFFIX,
C    A MATCH HAS BEEN MADE.
C
      IF(MSUFF2(NOSUFC+1).NE.IBLANK) GO TO 31700
      GO TO 31800
C
C31600* THE POTENTIAL SUFFIX IS LESS THAN THE TABLE ENTRY
C
C
31600 IHIGH2 = ICHCK2 - 1
      GO TO 31400
C
C31700*  THE POTENTIAL SUFFIX IS GREATER THAN THE TABLE ENTRY
C
31700 ILOW2 = ICHCK2 + 1
      GO TO 31400
C
C31800*  SUFFIX 2 HAS BEEN MATCHED.  CHECK EXCEPTION TABLE.
C
31800 IF(ILETCT(ICHCK2).EQ.0) GO TO 32200
      IF(K.EQ.1) GO TO 31900
      LSTCHR = KWORDI(K-1)
      GO TO 32000
31900 LSTCHR = IRTSAV(3)
32000 LBOT = ILETIN(ICHCK2)
      LTOP = LBOT + ILETCT(ICHCK2) - 1
      DO 32100 L2 = LBOT,LTOP
      IF(LSTCHR.EQ.LETXCP(L2)) GO TO 32200
32100 CONTINUE
```

```
C
C  THE LETTER RULE APPLIES, BUT NO MATCH WAS MADE
C
      GO TO 34000
C
C32200*  SEARCH THE EXCEPTION TABLE
C
C  RECONSTRUCT TEXT WORD 1.
32200 IF(IEXPCT(ICHCK2).EQ.0) GO TO 33500
      DO 32300 L3=1,3
      IWRDCK(L3) = IRTSAV(L3)
32300 CONTINUE
      IF(K.EQ.1) GO TO 32450
      LTEMP3 = K - 1
      DO 32400 L3 = 1,LTEMP3
      IWRDCK(L3 + 3) = KWORDI(L3)
32400 CONTINUE
32450 IWRDCK(K+3) = IBLANK
      IXCPT = IEXPIN(ICHCK2)
      ILOW3 = IXCPT
      IHIGH3 = IXCPT + IEXPCT(ICHCK2) - 1
32500 IF(ILOW3.GT.IHIGH3) GO TO 33500
      ICHCK3 = (ILOW3 + IHIGH3)/2
      LENEXP = IWRDXL(ICHCK3)
      ICHAR = (LENEXP+5)/6
      CALL BRKUP(IEXCPT,IWORDX(1,ICHCK3),ICHAR)
      DO 32600 L3 = 1,LENEXP
      JTEST = LCOMP(IWRDCK(L3),IEXCPT(L3))
      IF(JTEST) 32700,32600,32800
32600 CONTINUE
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE EXCEPTION
C    LIST ENTRY.  IF NO MORE CHARACTERS REMAIN IN THE TEXT WORD, A
C    MATCH HAS BEEN MADE.
C
      IF(IWRDCK(LENEXP+1).NE.IBLANK) GO TO 32800
C
C  THE WORD IS IN THE EXCEPTION LIST, THUS THE TEXT WORDS HAVE
C    DIFFERING STEMS.
      GO TO 34000
C
C32700*  THE TEXT WORD IS LESS THAN THE TABLE ENTRY.
C
32700 IHIGH3 = ICHCK3 - 1
      GO TO 32500
C
C32800*  THE TEXT WORD IS GREATER THAN THE TABLE ENTRY.
C
32800 ILOW3 = ICHCK3 + 1
      GO TO 32500
C
C33500*  THE TEXT WORDS HAVE THE SAME STEM.
C
33500 IRMATC(JTEMP) = MATNUM
C
C34000*  END OF J LOOP
C
34000 CONTINUE
C
C  INCREMENT MATCH NUMBER
```

```
      MATNUM = MATNUM + 1
C
C35000*  END OF I LOOP
C
35000 CONTINUE
      RETURN
C * * *
34500 WRITE (6,34600)
34600 FORMAT (1X,,TO BE CHANGEDC,)
      RETURN
C
      ENTRY SFMTC1
      READ (30)        NSUFF1,NSUFF2,NEXCEP,NLETTR
      MATNUM = 1
      READ (30)       ((ISUFF1(I,J),I=1,JWORDS),J=1,NSUFF1)
      READ (30)       (NOSUF1(J),INDSF2(J),ISFCN2(J),J=1,NSUFF1)
      READ (30)       ((ISUFF2(I,J),I=1,JWORDS),J=1,NSUFF2)
      READ (30)       (NOSUF2(J),IEXPIN(J),IEXPCT(J),ILETIN(J),ILETCT(J),
     1  J=1,NSUFF2)
      READ (30)       ((IWORDX(I,J),I=1,JWORDS),J=1,NEXCEP)
      READ (30)       (IWRDXL(J),J=1,NEXCEP)
      READ (30)       (LETXCP(J),J=1,NLETTR)
      RETURN
      END
```

```
CSFMATC        SFMATC--MODULE TO CALCULATE MATCH COUNTS
      SUBROUTINE SFMATC
C * * * * * * * * * * *
C
C            THIS VERSION OF SFMATC EMPLOYS THE SUBROUTINES KOR21A AND
C             KOR09A WHICH ALLOW CHARACTER MANIPULATION ON-PACKED STRINGS
C
C * * * * * * * * * * *
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,CDUMMY(3)
      COMMON /IMAT/ IRWORD(3,500),ICOUNT(500),IRMATC(500),IRTSAV(3)
      DIMENSION ISUFF1(3,150),NOSUF1(150),INDSF2(150),ISFCN2(150),
     1  ISUFF2(3,375),NOSUF2(375),IEXPIN(375),IEXCPT( 3),IEXPCT(375),
     2  IWRDXL( 800),IWORDX(3, 800),ILETIN(375),ILETCT(375),LETXCP( 50),
     3  IWRDCK( 3),KWORDI( 3),KWORDJ( 3),MSUFF1( 3),MSUFF2( 3),
     4  NTEMP(3)
      EQUIVALENCE (ILOW1,ILOW2,ILOW3),(IHIGH1,IHIGH2,IHIGH3),
     1  (ICHCK1,ICHCK2,ICHCK3),(NOSUFC,LENEXP)
      DATA IBLANK/6H      /,JWORDS/3/,MATNUM/1/
      DATA LSTCHR/1H /
C  CALCULATE-MATCH-COUNT-30000 SECTION.  CHECKS WORDS THAT HAVE BEEN
C    PLACED IN THE ROOT CHECK TABLE FOR COMMON ROOTS OR STEMS.  TOKENS
C    WITH MATCHING STEMS ARE ASSIGNED IDENTICAL MATCH COUNTS.
C
C
      IF(ISAVEC.NE.1) GO TO 10400
      IRMATC(1) = MATNUM
      MATNUM = MATNUM + 1
      RETURN
C
10400 DO 15 I=1,ISAVEC
  15  IRMATC(I) = 0
      ITEMP = ISAVEC
30000 DO 35000 I=1,ISAVEC
C  I IS THE INDEX FOR WORD 1.
C  LOOK FOR WORD1 (I) THAT HAS NOT BEEN ASSIGNED A MATCH COUNT
C
      IF(IRMATC(I).NE.0) GO TO 35000
C
C  ASSIGN MATCH COUNT TO WORD1.
C
      IRMATC(I) = MATNUM
C
      ITEMP = ITEMP - 1
      JTEMP = ISAVEC + 1
      DO 34000 J = 1,ITEMP
C
C  START AT BOTTOM OF TABLE AND LOOK FOR A WORD THAT HAS NOT BEEN
C   ASSIGNED A MATCH COUNT.
C
      JTEMP = JTEMP - 1
      IF(IRMATC(JTEMP).NE.0) GO TO 34000
C
C
C  CHECK FOR POINT AT WHICH THE TWO WORDS DIFFER
C
      CALL KOR21A(NSTEM,IRWORD(1,I),1,IRWORD(1,JTEMP),1,IDIFFR,IERR)
30050 IF(IDIFFR.NE.0) GO TO 30100
      IRMATC(JTEMP) = MATNUM
```

```
      GO TO 34000
C
C30100*  POINT OF DIFFERENCE HAS BEEN LOCATED.  THE REMAINDER OF BOTH
C   WORDS WILL BE TREATED AS POTENTIAL SUFFIXES
C
30100 IDIFFR = IABS(IDIFFR)
      KARG = (NSTEM + 1) - IDIFFR
      DO 30200 L = 1,3
      MSUFF1(L) = IBLANK
      MSUFF2(L) = IBLANK
30200 CONTINUE
      CALL KOR09A(KARG,IRWORD(1,I),IDIFFR,MSUFF1(1),1,IERR)
      CALL KOR09A(KARG,IRWORD(1,JTEMP),IDIFFR,MSUFF2(1),1,IERR)
C  THE POTENTIAL SUFFIXES HAVE BEEN TRANSFERED TO WORK AREAS
C
C
C  THE SMALLER SUFFIX (BY COLLATING SEQUENCE) SHOULD BE IN HOLD AREA 1.
C    IF IT IS NOT, MAKE A TRANSFER.  LCOMP IS A SUBROUTINE THAT DOES A
C    LOGICAL COMPARE (I.E. BY COLLATING SEQUENCE) OF TWO COMPUTER
C    WORDS
C
      CALL KOR21A(KARG,MSUFF1(1),1,MSUFF2(1),1,JTEST,IERR)
      IF(JTEST) 30700,34500,30300
C
C30300*  REMAINDER OF WORD1 IS HIGHER IN COLLATING SEQUENCE THAN
C    REMAINDER OF WORD 2
C
30300 DO 30400 N1=1,3
      NTEMP(N1) = MSUFF1(N1)
      MSUFF1(N1) = MSUFF2(N1)
      MSUFF2(N1) = NTEMP(N1)
30400 CONTINUE
C
C30700*  DO BINARY SEARCH FOR SUFFIX 1
C
30700 ILOW1 = 1
      IHIGH1 = NSUFF1
30800 IF(ILOW1.GT.IHIGH1) GO TO 34000
      ICHCK1 = (ILOW1 + IHIGH1)/2
      NOSUFC = NOSUF1(ICHCK1)
      CALL KOR21A(NOSUFC,MSUFF1(1),1,ISUFF1(1,ICHCK1),1,JTEST,IERR)
      IF(JTEST) 31100,30900,31200
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE SUFFIX IN
C    TABLE 1.  IF NO MORE CHARACTERS REMAIN IN THE POTENTIAL SUFFIX,
C    A MATCH HAS BEEN MADE.  OTHERWISE, THE POTENTIAL SUFFIX IS LARGER
C    THAN THE TABLE ENTRY
C
30900 NOSUFC = NOSUFC + 1
      CALL KOR21A(1,MSUFF1(1),NOSUFC,IBLANK,1,JTEST,IERR)
      IF(JTEST) 31200,31300,31200
C
C31100*  THE POTENTIAL SUFFIX IS LESS THAN THE TABLE ENTRY
C
31100 IHIGH1 = ICHCK1 - 1
      GO TO 30800
C
C31200*  THE POTENTIAL SUFFIX IS GREATER THAN THE TABLE ENTRY
C
31200 ILOW1 = ICHCK1 + 1
```

```
      GO TO 30800
C
C
C31300*  SUFFIX 1 HAS BEEN MATCHED.  CHECK FOR SUFFIX 2.
C
31300 ILOW2 = INDSF2(ICHCK1)
      IHIGH2 = INDSF2(ICHCK1) + ISFCN2(ICHCK1) - 1
31400 IF(ILOW2.GT.IHIGH2) GO TO 34000
      ICHCK2 = (ILOW2 + IHIGH2)/2
      NOSUFC = NOSUF2(ICHCK2)
      CALL KOR21A(NOSUFC,MSUFF2(1),1,ISUFF2(1,ICHCK2),1,JTEST,IERR)
      IF(JTEST) 31600,31500,31700
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE SUFFIX IN
C    TABLE 2.  IF NO MORE CHARACTERS REMAIN IN THE POTENTIAL SUFFIX,
C    A MATCH HAS BEEN MADE.
C
31500 NOSUFC = NOSUFC + 1
      CALL KOR21A(1,MSUFF2(1),NOSUFC,IBLANK,1,JTEST,IERR)
      IF(JTEST) 31700,31800,31700
C
C31600* THE POTENTIAL SUFFIX IS LESS THAN THE TABLE ENTRY
C
C
31600 IHIGH2 = ICHCK2 - 1
      GO TO 31400
C
C31700*  THE POTENTIAL SUFFIX IS GREATER THAN THE TABLE ENTRY
C
31700 ILOW2 = ICHCK2 + 1
      GO TO 31400
C
C31800*  SUFFIX 2 HAS BEEN MATCHED.  CHECK EXCEPTION TABLE.
C
31800 IF(ILETCT(ICHCK2).EQ.0) GO TO 32200
      IF(IDIFFR.EQ.1) GO TO 31900
      K2 = IDIFFR - 1
      CALL KOR09A(1,KWORDI(1),K2,LSTCHR,1,IERR)
      GO TO 32000
31900 CALL KOR09A(1,IRTSAV(1),3,LSTCHR,1,IERR)
32000 LBOT = ILETIN(ICHCK2)
      LTOP = LBOT + ILETCT(ICHCK2) - 1
      DO 32100 L2 = LBOT,LTOP
      LETHLD = ILETIN(ICHCK2) + L2 - 1
      IF(LSTCHR-LETXCP(L2)) 32100,32200,32100
32100 CONTINUE
C
C  THE LETTER RULE APPLIES, BUT NO MATCH WAS MADE
C
      GO TO 34000
C
C32200*  SEARCH THE EXCEPTION TABLE
C
C  RECONSTRUCT TEXT WORD 1.
32200 IF(IEXPCT(ICHCK2)) 32300,33500,32300
32300 CALL KOR09A(3,IRTSAV,1,IWRDCK(1),1,IERR)
      IF(IDIFFR-1) 32400,32450,32400
32400 LTEMP3 = IDIFFR - 1
      CALL KOR09A(LTEMP3,KWORDI(1),1,IWRDCK(1),4,IERR)
32450 K1 = IDIFFR + 1
```

```
      CALL KOR09A(1,IBLANK,1,IWRDCK(1),K1,IERR)
      IXCPT = IEXPIN(ICHCK2)
      ILOW3 = IXCPT
      IHIGH3 = IXCPT + IEXPCT(ICHCK2) - 1
32500 IF(ILOW3.GT.IHIGH3) GO TO 33500
      ICHCK3 = (ILOW3 + IHIGH3)/2
      LENEXP = IWRDXL(ICHCK3)
      CALL KOR21A(LENEXP,IWRDCK(1),1,IWORDX(1,ICHCK3),1,JTEST,IERR)
      IF(JTEST) 32700,32600,32800
C
C  THE CHARACTERS WERE EQUAL THROUGH ALL THE LETTERS IN THE EXCEPTION
C    LIST ENTRY.  IF NO MORE CHARACTERS REMAIN IN THE TEXT WORD, A
C    MATCH HAS BEEN MADE.
C
32600 LENEXP = LENEXP + 1
      CALL KOR21A(1,IWRDCK(1),LENEXP,IBLANK,1,JTEST,IERR)
      IF(JTEST) 32800,32650,32800
C
C  THE WORD IS IN THE EXCEPTION LIST, THUS THE TEXT WORDS HAVE
C    DIFFERING STEMS.
32650 GO TO 34000
C
C32700*  THE TEXT WORD IS LESS THAN THE TABLE ENTRY.
C
32700 IHIGH3 = ICHCK3 - 1
      GO TO 32500
C
C32800*  THE TEXT WORD IS GREATER THAN THE TABLE ENTRY.
C
32800 ILOW3 = ICHCK3 + 1
      GO TO 32500
C
C33500*  THE TEXT WORDS HAVE THE SAME STEM.
C
33500 IRMATC(JTEMP) = MATNUM
C
C34000*  END OF J LOOP
C
34000 CONTINUE
C
C  INCREMENT MATCH NUMBER
      MATNUM = MATNUM + 1
C
C35000*  END OF I LOOP
C
35000 CONTINUE
      RETURN
C * * *
34500 WRITE (6,34600)
34600 FORMAT (1X,,TO BE CHANGEDC,)
      RETURN
C
      ENTRY SFMTC1
      REWIND 30
      READ (30)       NSUFF1,NSUFF2,NEXCEP,NLETTR
      MATNUM = 1
      READ (30)      ((ISUFF1(I,J),I=1,JWORDS),J=1,NSUFF1)
      READ (30)      (NOSUF1(J),INDSF2(J),ISFCN2(J),J=1,NSUFF1)
      READ (30)      ((ISUFF2(I,J),I=1,JWORDS),J=1,NSUFF2)
      READ (30)      (NOSUF2(J),IEXPIN(J),IEXPCT(J),ILETIN(J),ILETCT(J),
```

```
     1  J=1,NSUFF2)
        READ (30)       ((IWORDX(I,J),I=1,JWORDS),J=1,NEXCEP)
        READ (30)       (IWRDXL(J),J=1,NEXCEP)
        READ (30)       (LETXCP(J),J=1,NLETTR)
        RETURN
C
C
        END
```

```
CSFINAL         SFINAL--FINAL OUTPUT MODULE
C
C                SUBROUTINE SFINAL POST-PROCESSES FUNCTION WORDS AND
C         AND PUNCTUATION RECORDS, AND PRINTS A SUMMARY.  THE
C         INDIVIDUAL FUNCTION AND PUNCTUATION TOKEN RECORDS ARE FOUND
C         ON SCRATCH FILE 18.  THE TYPE RECORDS ARE FOUND ON SCRATCH
C         FILE 17.
C
      SUBROUTINE SFINAL(NEWPAG,IPRINT)
      COMMON /ALL/ NOCHAR,NCHARP,NSTEM
      COMMON /RPRT/ ITITLE(14),KDUMMY(143)
      COMMON /CNTR/ INCNT,KFUNCT,KPUNCT,KCURR,ISAVEC,JFNFLG,JPNFLG,
     1   ICNTNT
      DIMENSION INWORD(18),INPFIX(8),INDEXN(8),MATTOT(2)
      DATA IBLANK,IDUM/1H ,0/,LEOF,LEXCLM/077202020202020,6HEXCLAM/

      IF(JPNFLG.NE.1.AND.JFNFLG.NE.1) GO TO 78000
      IF(IPRINT.NE.1) GO TO 70100
      LABEL1 = 1
      GO TO 76000
70000 WRITE (17) IDUM,IDUM
      REWIND 17
      REWIND 18
C       INITIALIZE DATA ARRAYS
      MATTOT(1) = KPUNCT
      MATTOT(2) = KFUNCT
70100 READ (17) KCURR,IFNTYP
      IF(KCURR.EQ.0) GO TO 78000
      READ (18) JMATOT,INPRFL,INWRDL,INPFIX,INWORD
      DO 70200 K=1,KCURR
      READ (18) INDEXN
      WRITE (13) INDEXN,JMATOT,MATTOT(IFNTYP),KCURR,INPRFL,INWRDL,
     1  INPFIX,INWORD
      JOTCNT = JOTCNT + 1
70200 CONTINUE
      IF(IPRINT.NE.1) GO TO 70100
C
C       NEW RECORD TYPE
C
      IF(LINES.LT.NEWPAG) GO TO 70800
      LABEL1 = 2
      GO TO 76000
C       WRITE AUDIT REPORT RECORD
70800 IF(INWORD(1).EQ.LEOF) GO TO 72000
      WRITE (6,70900) INWORD,KCURR
70900 FORMAT(1X,18A1,5X,I4)
      LINES = LINES + 1
      GO TO 70100
C         WRITE OUT EXCLAMATION RECORD TO AVOID SLEW CHARACTER
72000 WRITE(6,72100) LEXCLM,IBLANK,IBLANK,KCURR
72100 FORMAT (1X,3A6,5X,I4)
      LINES = LINES + 1
      GO TO 70100
C
C       NEW PAGE ROUTINE
C
76000 IPAGE = IPAGE + 1
      LINES = 1
      WRITE (6,76100) ITITLE,IPAGE
```

```
76100 FORMAT(1H1,13A6,A2,31X,5HPAGE ,I4)
      WRITE (6,76200)
76200 FORMAT(1HO/30X,42HFUNCTION WORDS AND PUNCTUATION --- SUMMARY)
      LINES = LINES + 3
C       IS THIS THE FIRST PAGE
      IF(ICONT.EQ.1) GO TO 77000
      IF(JPNFLG.EQ.0) GO TO 76400
      WRITE (6,76300) KPUNCT
76300 FORMAT (1HO,18X,11HTHERE WERE ,I5,49H PUNCTUATION RECORDS OUTPUT (
     1MATCH COUNT = 99999))
      LINES = LINES + 2
76400 IF(JFNFLG.EQ.0)GO TO 76800
      IF(JPNFLG.EQ.0) GO TO 76600
      WRITE (6,76500)
76500 FORMAT (1HO,49X,3HAND)
      LINES = LINES + 2
76600 WRITE (6,76700)KFUNCT
76700 FORMAT (1HO,18X,11HTHERE WERE ,I5,46H FUNCTION RECORDS OUTPUT (MAT
     1CH COUNT = 99998))
      LINES = LINES + 2
76800 ICONT = 1
      GO TO 77200
77000 WRITE (6,77100)
77100 FORMAT (1HO,45X,11H(CONTINUED))
      LINES = LINES + 2
77200 WRITE (6,77300)
77300 FORMAT (1HO,4X,4HWORD,13X,10HTYPE-COUNT///)
      LINES = LINES + 5
      GO TO (70000,70800),LABEL1
C
C       END OF TOKEN RECORD FILE
C
78000 IPAGE = IPAGE + 1
C
C         WRITE DUMMY RECORD FOR END OF FILE
C
      WRITE (13) (IDUM,J=1,13),(LEOF,J=1,26)
      WRITE (6,76100) ITITLE,IPAGE
      KTOTAL = JOTCNT + ICNTNT
      WRITE (6,78100) INCNT,JOTCNT,ICNTNT,KTOTAL
78100 FORMAT (1HO/X,19HRECORD COUNTS -----/24X,20HTOTAL RECORDS INPUT ,
     1  I5/4X,40HFUNCTION AND PUNCTUATION RECORDS OUTPUT ,I5/16X,
     2  28HCONTENT WORD RECORDS OUTPUT ,I5/23X,21HTOTAL RECORDS OUTPUT ,
     3  I5/////30X,22H***** END OF JOB *****)
      STOP
      END
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
University of Kansas Lawrence, Kansas 66044	Unclassified
	2b. GROUP

3. REPORT TITLE

Automated Analysis of Language Style and Structure in Technical and Other Documents

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Sedelow, Sally Yeates

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
1 September 1971	275	

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
N00014-70-A-0357-0001	
b. PROJECT NO.	
c.	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Distribution of this report is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The project concerned with Automated Analysis of Language Style and Structure in Technical and Other Documents has had four primary emphases during the 1970-71 research year. One emphasis has been the development of approaches to the modeling of thesauri, so that existing thesauri can be precisely described and new thesauri (more adequate to a wide range of information retrieval requirements) can be developed. A second emphasis has been the writing of statistical data-gathering programs and the testing of statistical analytical tools on that data; this work is directed toward a statistical support package for natural language research. A third emphasis has been upon beginning the development of a theory of prefixing in English in order to provide a basis for automated handling of prefixes. The fourth emphasis has been upon programming a FORTRAN IV, Honeywell 635 version of the list-structure VIA programs, which are used for content analysis as well as for inputs to the research on thesauri and statistical analysis.

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Automated Language Analysis						
Stylistic Analysis						
Thesauri						
Prefixing						
Content Analysis						
Statistical Package						