II. THE SMART SYSTEM - GENERAL PROGRAM DESCRIPTION

Michael Lesk

1. Introduction

SMART (Salton's Magical Automatic Retriever of Texts) is a programming system designed to help evaluate proposed information retrieval systems. It includes many different methods for analyzing a collection of English documents and answering search requests.

Sets of input documents and requests are processed in two steps. Step one involves the reduction of the document to a form useful for machine analysis, including its representation by a list of "concepts" and associated "weights". If all knowledge is assumed to be representable as a kind of n-dimensional space, any piece of information in the text may be found by a displacement within this space in some given direction. The different directions are identified by "concept numbers," of which up to 1,000 may be defined; each document is then represented by a vector in the space of concepts. As a low order approximation, concepts may be equated with text words. The programs incorporated into SMART provide, however, many ways of eliminating or mitigating the ambiguities and duplications inherent in this approximation. The weights attached to the concepts may merely reflect the frequency of occurrence of the concepts, or can be adjusted in several ways.

The representation of a document by a concept vector permits step two of the retrieval system to operate. This step compares vectors representing retrieval requests (whose treatment is in general identical with that of reference documents) with document vectors, and produces lists of "answers." Should results prove unsatisfactory, various options permit systematic alterations of the request to produce different answers.

The value of the system lies in the variety of methods that may be used at each step. The methods include many of the schemes that have appeared in the literature and thus make SMART particularly useful for comparing such proposals.

In step one, the reduction of English text to concept-vector format, the programs must first read the text and convert text words to concept numbers by reference to a dictionary. If one wishes to treat each distinct English word as a distinct concept, one supplies the program with a "null" thesaurus assigning to each word a distinct number. Such a thesaurus may be generated automatically. If, on the other hand, one wishes to consider _a priori_ relations believed to exist between words, one must of course supply the program with a hand-constructed dictionary. This dictionary may assign "orthogonalized" concept numbers, or, alternatively include complex interrelationships. Each word in the SMART dictionary, or thesaurus, may map into six different concepts, and each concept can represent any number of words. Phrase searching procedures are used to group any collection of specified words and syntactic relations into a single concept, and concept relationships may be stipulated in a hierarchical form. Alternatively, concept relationships may be obtained from statistical criteria, which make no assumptions about the _a priori_ meanings of words. In fact, except for keypunching problems, many of the SMART options could function for foreign language texts, without requiring the researcher to know anything about that language.

Request answering may also be handled in many ways. The fundamental procedure is to compare requests with documents by one of three correlation methods. If this proves unsatisfactory, however, concept vectors can be altered in several ways. Statistical criteria can be applied to detect relationships between concepts, or _a priori_ hierarchical relationships may be given in tree structure form. Such a tree structure may be used to transform the concept vectors in several ways, all alterations of requests are followed by a recomputation of the correlations and a new analysis of request answers.

The two-step process can thus be visualized as consisting of intra-document and interdocument operations. When all intradocument processing is completed, the resulting concept-vector may be punched into binary cards. These binary cards, called the document "list" of concept occurrences, can be read into the machine 100 times faster than the unprocessed alphabetic text, and can then be used for all interdocument operations.

Various types of concept vectors may be produced by the intradocument processing depending on the thesaurus and the processing options which are called into play. For example, if "orthogonalized" concepts are used, in which each piece of information has unique representation as a concept vector, correlations may be performed immediately. Alteration of the request vector would necessarily change the meaning of the request and might then be of little direct usefulness. On the other hand, if a null thesaurus is used with no phrase-finding options, each text word will have been replaced by a distinct numeric code. Additional processing of the resulting concept vectors might in this case be quite profitable. Processing options are discussed in detail in the other sections of this report.

The programmed version of SMART runs on the IBM 7094 under a FORTRAN II monitor. Twelve tapes are required if all options are used: A1-6 and B1-6 (including FMS tapes). 32K core is essential. The Harvard monitor system is an almost standard FORTRAN II system tape, but persons desiring to run elsewhere should read the section on "Operational Details" carefully.

SMART is a chain job, with up to seven processing links, plus two links for writing a library tape with the necessary dictionaries. The links are numbered 11,10,1,2,3,4,5,6,7. The preprocessor which writes the library tape, consisting of five files is programmed as links 11 and 10. The first link, 11, writes the first file of the library, which includes the thesaurus and its suffix lists. Link 10 adds four more files, which are (in order), another suffix list, a dictionary of phrases specified by syntactic/semantic/structural information, a dictionary of phrases specified only by semantic information, and the tree structure (hierarchy) of concepts. Once generated, the library tape may be mounted on the proper tape drive and the first two program links omitted.

After the library is written, control passes to link 1, the main link in SMART. This link contains all request answering programs, the concept vectors for all documents, all statistical programs, the system supervisor, and much else. Logically, all other links are merely long subroutines called by it. Link 2 performs dictionary lookups; during its execution, link 1 is saved on a scratch tape. Links 3-7 perform syntactic analysis of texts and phrase searching, using phrases specified syntactically and semantically. In this five-link section, links 4, 5, and 6 contain the Kuno Multiple Path Syntactic Analyzer, described in The Computation Laboratory of Harvard University Reports NSF-8 and 9. Link 3 is a pre-editor for the analyzer. Link 7 compares the syntactically analyzed sentences to the library of phrases, using the Sussenguth structure-matching programs (see Sec. VIII of this report). The remainder of link 7 handles format conversions and bookkeeping. The main program is again on a scratch tape during these operations. The figures show the flow of data from tape to core in the various links, as well as the compiled tape assignments. On the Harvard system tape, A1 contains the FORTRAN monitor, A2 is the card input tape, A3 the off-line print and B4 the off-line punch tape.

2. The SMART Library Tape

The five files of the SMART library tape, as previously indicated, are written by the first two chain links of the program. The first file, which is used for each text or request submitted in English, contains the thesaurus. This thesaurus is used by chain link 2, the lookup, in translating English into concept numbers. The first file contains four records: two label records, the thesaurus, and a suffix tree used to assist the thesaurus. The only function of chain link 11 is to write the thesaurus file; the corresponding program deck should be the first binary deck in the job if a new library is to be written. It exits to chain link 10, which writes the remaining four files. The second file contains another suffix list, used to obtain syntactic information in link 3, the preprocessor for the syntactic analyzer. This file is written by subroutine UPSUFF. UPCRIT is the next subroutine called; this routine writes the dictionary of criterion trees onto the library as the third file; each tree appears as a separate record. The criterion trees further described in a later section by Lemmon define a phrase by its syntactic, semantic, and structural content. Chain link 7 will compare this file with the syntactically analyzed sentences to detect phrases. Another, less accurate, phrase dictionary is written as file number four by subroutine CRITS2, in records of 100 phrases. In this file, a phrase is defined only by its constituent semantic concepts, without syntactic or structural information. A quick, rough search can thus be made by subroutine PHROCC to detect phrases, without calling on the syntactic analysis programs. The fourth file is also used by the syntactic routines for additional information about the syntactic phrases which are found. Finally, the fifth file, written by subroutine GEORGE, contains the concept hierarchy. All concept vector expansions through the hierarchy use this file, which consists of two records (see Sec. V of this report).

In general, each of the update subroutines expects to find an old library tape on A6, and correction cards on A2 (system input tape), before writing a new file in B5, and spacing A2 and A6 forward to the next file. In order to get started, options are provided for writing files entirely from cards. If a subroutine detects an error in the correction cards, an indication of this is given on the print tape and sense light one is turned on as a signal to the update main program. Execution is terminated after updating if sense light one is found on. Otherwise, the tapes are rewound and the run proceeds as if a library tape had been premounted on B5.

In using a null dictionary, there is no need for syntactic tables or hierarchies; only the dictionary is of significance. To avoid the unnecessary writing of all five files, chain link 10 may be omitted from a run and link 11 may be used alone. In this case (since link 11 will try to call link 10 at exit), the main link (usually link 1) may be renamed link 10. This requires only the change of the * CHAIN(1,4) card to * CHAIN(10,4) and will not disturb any other part of the system. Alternatively, a dummy link 10 might be supplied.

It should be noted that A6 is used both as an intermediate tape during syntactic processing, and as the unit for the old library tape in updating. Thus, if syntactic processing is done in the same run with updating, the old tape may not be file protected. Care should be exercised in using this combination of options.

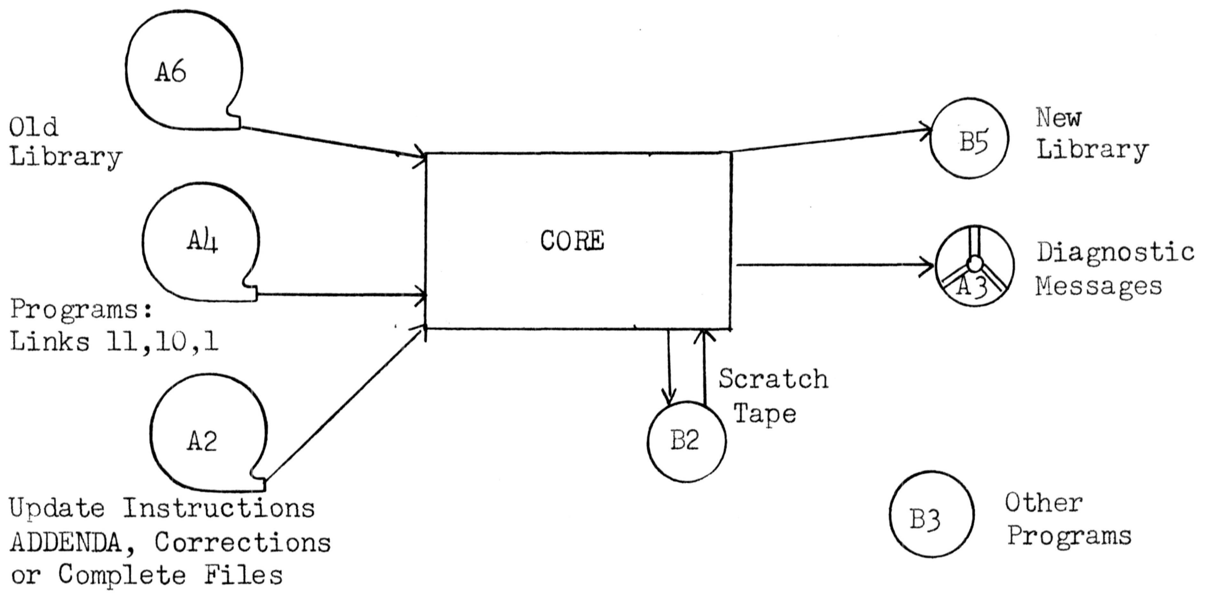
Summary of the SMART Library Tape (See Fig. 1)

File 1. Four records: one-word label, thesaurus (approximately six machine words per entry), one-word label, suffix list.

File 2. One record. Suffix list with syntactic information. Approximately 255 machine words.

File 3. One record per criterion tree. Criterion trees with syntactic/semantic/structural specifications.

For simple trees, about 10-20 machine words per tree.



Link		Output Library
11	File 1: Four Records	Thesaurus (English ⟶ concept numbers + syntax codes) Suffix list (suffixes ⟶ suffix numbers).
10	File 2: One Record	Suffix list (suffix numbers ⟶ syntactic codes).
	File 3: One Record/ Phrase	Criterion trees for phrases. Complete semantic/syntactic/structural phrases.
	File 4: One Record/ 100 Phrases	Statistical phrase search data. Phrases with only semantic specification.
	File 5: One Record	Hierarchy. Concept tree structure, with crossreferences.

Dictionary Update System

Figure 1

File 4. 100 phrases per record. Four words per phrase. The last record is short. All records have a three word label in front of them. Semantic phrase data.

File 5. Two records. Five-word label; then hierarchy and cross-references.

3. The CHIEF Supervisor

After the optional updating is completed, general control of the SMART system passes to link 1. This link contains all data accumulated for documents in each run, all statistical and hierarchical programs, the request answering programs, and many general housekeeping routines. Link 1 reads all input cards, writes all of the punch output and most of the print output.

The most inportant program in this link is the supervisory program called CHIEF. Its function is to call the subroutines responsible for the various processing options as needed, and to see to it that all necessary tables, flags, and constants are kept up to date. CHIEF makes its decisions for processing on the basis of the current input on A2 and the options chosen at the beginning of the run.

Three main phases of the program are recognized. The first is a short initialization stage. Its major task is to read the list of options and parameters for this run from the input tape, and to decode it. Various tables and constants are also read in and set up internally. The second stage is devoted to absorbing the document collection for this run. The mixture of English texts and preprocessed documents is read in and processed as required by the preset options. When the whole collection is present in concept-vector form, CHIEF passes to the third stage, that of retrieval operations. The programmer may choose any of the various retrieval options available, or omit this stage entirely. The three stages are described in the next several paragraphs.

A. PHASE I: Initialization

Before the required options can be read in, some housekeeping must be done. Numerous flags are set equal to zero, lights turned off, and tapes rewound. The current time is recovered through CLOCK, the date through DATEIS, and the first page heading is written (by HEADR). The most significant call is to TAPENO to set up the tape instructions in all subroutines. When these instructions are set, the program can proceed to read the input tape. Subroutine NAMES reads the first batch of data consisting of the list of identifiers for the concept categories. This list, stored in array RONAME, is used by the program whenever a concept number must be referred to; the six character name, rather than the number itself, is always printed.

CHIEF now calls subroutine SETFLG to perform the major task of this phrase, the decoding of the control cards specifying all the processing options. These control cards follow the list of concept names (or the * DATA or library cards if the optional names list is omitted). The control cards must precede all documents.

Each specification is represented by a string of up to 24 BCD characters plus blanks as desired by the programmer, and separated by a comma from any other specifications. There exist four classes of specifications, as follows:

(1) Instructions to set a bit in a control word called FLAG (octal 77461) corresponding to a yes-no processing choice; a flag bit would thus control output printing of the list of words not found in dictionary, or the mode of concept expansion (addition or replacement of terms).

(2) Instructions to set a program parameter to some numerical or logical value. A typical numerical parameter would specify the value of a cutoff in some correlation procedure; a typical logical parameter might correspond to the method of hierarchial expansion (sons, parents, brothers, or cross references).

(3) One specification that writes an identifying card on the punch tape.

(4) Erroneous specifications, which produce nasty messages on the print tape.

Type 2 specifications are punched as a string of characters giving first the six character parameter name, and then its desired value. The whole string may be prefixed with SET PARAMETER if a more readable output is wanted. For example, to specify the cosine correlation mode, one punches CORMOD COS or SET PARAMETER CORMOD COS. To set the maximum concept number to 374, one writes MAXCON 374 or SET PARAMETER MAXCON 374. At present there exist 35 type 1 specifications, ten of type 2, and one of type 3. A debugging specification, also of type 1, raises the total to 47. Each type 1 specification may be written in two forms, a long, readable form and a short, easy to keypunch form, adding up to a total of 94 legal specifications. A list of these options is given in Table 1; they are also further discussed in the text.

B. PHASE II: Document Absorption

In this phase CHIEF supervises the read-in and processing of all documents. The input tape consists of documents, in binary or BCD format, separated by instruction cards for CHIEF. These instruction cards are marked by an * in column one, and are not to be confused with the specifications to SETFLG discussed in a previous part of this section. CHIEF reads instruction cards until it finds one that refers to a document; at that point transfer is made to the appropriate subroutines for processing.

Each instruction card is identified by an * in column one, a four letter code in columns 2-5, and a blank in column 6. If the instruction refers to a document, the document is specified by a twelve-character name in columns 7-18. The remainder of the card is available for comments. The eight instructions are:

(1) *NOTE Comment card. Columns 7-72 printed, card ignored.

(2) *TIME The current time is printed.

(3) *STOP Begin phase III; no more documents on tape.

(4) *RAND DOCUMENT NAME Generate a pseudo document with random occurrences of random concepts and use it in all operations as if it were real. CHIEF calls an internal subroutine, NAMEIN, to enter the document name into the tables, and two external subroutines,

Long Form	Short Form Bit Affected in Flag	
Across the board	AB	All debugged bits
All analyses	AA	28
Answer requests	AR	8
Cluster concepts	KC	1
Cluster documents	KD	12
Clustering search	CS	22
Concept correlations	CC	3
Concept frequencies	CF	4
Concept relations	CR	2
Criterion trees	CT	27
Document correlations	DC	15
Document relations	DR	13
Edited Answers	EA	18
English texts	ET	35
Execute syntax	ES	19
Flip card XXXXXX	FC	XXXXXX See note following
Hierarchical expansion	HE	10
Hierarchical vectors	HV	11
Logical vectors	LV	21
Merged clustering	MK	17
Phrase frequencies	PF	16
Phrase search	PS	26
Punch document data	PU	23
Replace by hierarchy	RH	9
Replace by syntax	RS	24
Request correlations	RC	14
Set parameter	See note following	
Smear concepts	SC	7
Smear replacing	SR	6

Processing Specifications

TABLE 1

Long Form	Short Form Bit Affected in Flag	
Smeared vectors	SV	5
Statistical clusters	SK	30
Statistical trees	ST	25
Syntactic analysis	SA	29
Term correlations	TC	32
Term relations	TR	31
Texts processed	TP	20
Word frequencies	WF	33
Words not found	NF	34

The effect of the specifications above is generally given by imagining the long form prefixed by print. The exceptions are obvious.

The specification flip card does not cause any bits of flag to be set. Flip card causes the next six nonblank characters to be punched in a flip card. If six nonblank characters do not precede the next comma, blanks are used as fill. Do not use special characters other than the 8-5 punch, which is blanked. These flip cards are punched immediately.

Set parameter sets up constant for the program and sets mode mode switches for correlating and expanding. Type "Set parameter PARAMX value" or just "PARAMX value", where blanks are ignored and value is to be set as the value of PARAMX possibilities are:

<u>Parameter Values</u>

GOTREE	Root, branch, filial, refer
CORMOD	COS, ASYM, OVLAP

TABLE 1 (continued)

Parameter Values (continued)

CUTOF3	CUTOFF times 10,000 for document-document correlation
CUTOF1	Same for term-term correlation
CUTOF2	Same for concept-concept correlation
MAXCON	Highest thesaurus number

At least one specification that produces output should be given of course.

TABLE 1 (continued)

RANDOC (to generate the random document), and RELOC (to enter its concept vector into the document-concept matrix).

(5) *LIKE DOCUMENT NAME The document whose name is given is marked as a request, by internal subroutine NAMEIN. NAMEIN maintains tables of document names (NAME1 and NAME2) plus a table of flags for each document (DFLAG). The name specified on the *LIKE card is looked up in the tables, and the corresponding DFLAG entry is set to indicate a request.

(6) *LIST DOCUMENT NAME A document in binary form follows on the input tape. CHIEF calls successively: LISTIN to read the document, MERLST to prepare it for addition to the document concept matrix, and RELOC to perform the addition of the document to the collection.

(7) *FIND DOCUMENT NAME Equivalent to a *TEXT followed by a *LIKE, with minor differences. The most important difference is that a document looked up by a *FIND rather than a *TEXT will not be punched out in processed form; for this reason, the use of this instruction should be avoided.

(8) *TEXT DOCUMENT NAME Standard method of input for an English text. Initiates all processing for the addition of a document to the collection. The decoding of this instruction in accordance with the specifications from SETFLG is the major job of phase II and is described below.

Control passes first to subroutine SEGMNT to read in the text. If the ENGLISH TEXTS(ET) specification is given, sense light one is turned on to indicate printing of the input text. The text must now be looked up in the dictionary; this is done by link two, the lookup. Subroutine LINKUP, internal to CHIEF, dumps most of core onto A4 as one, self-loading record. Since this record is over 20,000 words long, tape writing errors are relatively frequent. Five consecutive attempts are therefore made to complete the read or write operations in case of error. Before calling link2, sense light four is turned on if the specification WORDS NOT FOUND (NF) was previously given, causing the printout of the words not found in the dictionary. Sense light three can also be turned on, telling the lookup to write a tape of syntactic information for link 3, the analyzer pre-processor. This tape writing is normally suppressed by means of the NO SYNTACTIC PROCESSING option (NS).

On return from the lookup, the program proceeds to the statistical processing of the text. Subroutine MATRIX tabulates the occurrences of concepts within sentences; this list is printed by subroutine WFREQ if

WORD FREQUENCIES (WF) is specified. ROWSUM and ROCOR now compute term-sentence correlations, giving a measure of similarity between any two concepts based on their co-occurrences in the same sentence. The correlation may be done in any of three ways, selected by the type two specification CORMD1. This specification can take the three logical values: COS, ASYM, OVLAP. These modes are further described as part of the statistical processing. Correlations may be printed by subroutines WCORR and WCORND if TERM CORRELATIONS (TC) is specified. MCORR and FRIEND are subroutines, not directly called by CHIEF, to convert the correlation vector into a list of concepts related to each concept in the text. This list is stacked in core (the actual correlations are not preserved), and the TERM RELATIONS specification causes WGROUP to print it. An attempt is now made to synthesize clusters of terms and count them as units. The subroutines involved are: CLUMP, CLSMAP and CDENSE, and the specifications involved are STATISTICAL CLUSTERS (SK) to print the clusters, and CLUSTERING SEARCH (CS) to initiate the entire operation. If abstracts or titles are processed, rather than complete documents, the statistical clustering procedure is inapplicable and NK is specified every run. A subroutine called NOCLMP handles housekeeping details normally taken care of by CLUMP.

The statistical clustering procedure is important even if not used, however, because of its effect on the processing mechanism. From this point on,the program must permit the handling of term clusters as well as single concepts; a new numbering scheme is therefore introduced. The document is represented in terms of intermediate "cluster numbers" which may stand for either a cluster or a single concept. A table called MAPKLS is used to store the data necessary to convert cluster numbers to concept numbers.

CHIEF now initiates the counting of the occurrences of the "clusters" (which may be single concepts) by calling KCOUNT. At this point one of two phrase detection systems may be called. The first is the full syntactic system using the Kuno multiple-path syntactic analyzer and the graph-matching routines of Sussenguth. The syntactic processing is called if EXECUTE SYNTAX (ES) is specified at option time. The first analysis produced by the Kuno analyzer is printed if option SYNTACTIC ANALYSIS (SA) is exercised; ALLANALYSES (AA) will print all syntactic analyses. CRITERION TREES (CT) prints the list of trees found to match any text excerpt. To enter the syntactic system, CHIEF dumps itself on A4, and calls in link 3 (BTOKUN) from B3. BTOKUN reads the syntactic information from tape A6 and prepares an input tape to the Kuno analyzer (using file 2 of the library on B5) on unit B6. Control is then turned over to link 4 (SETUP), the first part of package B of the Kuno analyzer. SETUP, using the condensed grammar tape on A5, prepares a binary sentence tape on B1. SYNTAX, link 5, is now executed; this produces a binary analysis tape on B2. Link 6, the final part of the Kuno analyzer (EDIT), writes a BCD output analysis tape on A6. Finally, link 7 compares this analysis tape with the criterion tree dictionary in file 3 of B5, and leaves a table of detected trees in core for CHIEF. A detailed description of this process is given as part of the syntactic section by Lemmon.

Alternatively, the cheap phrase finder may be used. This subroutine, PHROCC, uses the fourth file of the library tape, in which phrases are defined merely as a pair or triplet (up to sextet) of concepts co-occurring in a single sentence. PHRASE SEARCH (PS) specifies this search, and STATISTICAL TREES (ST) prints the results. This is described in a later section by Evslin and Lesk.

At this point each given document has undergone all the intra-document processing possible. A minimum of ten seconds is required for the lookup operation, and up to perhaps two minutes for the full syntactic processing. To save time on a subsequent run, each document can be punched out onto binary cards by LSTOUT. This punching is controlled by the PUNCH DOCUMENT DATA (PU) option. The deck punched in this operation may be resubmitted with a *LIST instruction in future runs. When a binary "list" of concept co-occurrences is used as input in this manner, LISTIN restores the internal concept vector as it existed previously at the same point. The SMART system no longer knows or cares, whether the document was read in using a *LIST or *TEXT control card. The programmer should be careful, however, to avoid mixing documents looked up in different thesauruses, as the concept numbers may not represent the same concepts. Each document concept vector is now added to the complete collection. Since term clusters may have been generated, the program first examines all "cluster numbers" to see whether they represent single concept numbers from the thesaurus, or groups of concepts combined by the clustering programs. True clusters (i.e. clusters of more than one concept) are compared with the clusters previously detected in the collection. If a cluster previously detected reappears, it is "merged" into the table of term clusters by identifying it with its previous version. Clusters are considered identical if they are "almost" the same members; "almost" being defined as

$$\frac{\text{number of common members}}{\sqrt{\text{product of size of each cluster}}} > 0.5$$

A new cluster which does not "match" any older cluster is added to the end of the combined cluster list. To avoid confusing such a cluster with a true thesaurus concept, it is assigned a "fake" concept number, higher than 1,000. An element of the concept-vector of a new document corresponding to a single thesaurus concept is entered directly into document-concept vector, as under its proper concept number. The bookkeeping needed for the comparison and counting operations is performed by subroutine MERLST.

Finally, RELOC and a subroutine called FILLIN, copy the processed document vector into the document-concept matrix, DOCWRD. Control now returns to the instruction card interpreter in CHIEF and phase II proceeds with the next document.

C. PHASE III: Request Answering

After the appearance of a *STOP card or an end of file, no further documents are read in and the processing of requests begins. A list of documents submitted during the present run can first be printed (using the TEXTS PROCESSED (TP) option), followed by a table of the merged statistical clustering, if any (using the MERGED CLUSTERING (MK) specification). The program then correlates the document concept matrix to get document-document correlations. These are obtained and printed for all documents if DOCUMENT CORRELATIONS (DC) is specified; documents whose correlation exceeds the cutoff are printed if DOCUMENT RELATIONS (DR) is given as an option. CLUSTER DOCUMENTS (KD) causes the program to cluster the documents into groups.

Normally, one is interested in the response of the system to retrieval requests. ANSWER REQUESTS (AR) causes the program to correlate the request vectors, and print all documents having correlations higher than the parameter CUTOF3. REQUEST CORRELATIONS (RC) will produce the correlation vector as output.

Two principal methods can now be used to alter the original search requests. The first of these consists in using concept-concept correlations; the second, hierarchical expansions. If concept-concept correlations are to be computed, the program first transposes the document-concept matrix by subroutine TRANS, to obtain a matrix useful for concepts rather than document processing. Subroutine CONCOR carries out the concept correlations using ROCOR and ROWSUM. The options CONCEPT CORRELATIONS (CC) and CONCEPT RELATIONS (CR) are used to print the corresponding data. The actual request alteration is performed by subroutine SMEAR, which adds to the original concepts of a request, new concepts which are related to them. This expansion is produced by the SMEAR CONCEPTS option (SC), and its results are printed by the SMEARED VECTORS (SV) specification. The requests are recorrelated using the altered concept vectors after this procedure, and new answers are produced. Programs are being written to provide for contraction as well as expansion of the search request using concept-concept correlation; options CONCEPT CLUSTERS (KC) , and REPLACE BY SMEARING (SR) control these alterations. The correlation mode and cutoff parameter are set by CORMD2 and CUTOF2 specifications of type 2. Additional processing details are included in the section on statistical processing.

Hierarchical expansion is performed through subroutine TREES. It is initiated by the option HIERARCHICAL EXPANSION (HE) and its details may be printed by means of the HIERARCHICAL VECTORS (HV) specification. Each document concept-concept vector is extracted from the document concept matrix, transmitted to the hierarchy programs (Sec. V) and returned to the matrix after alteration. The parameter GOTREE may be set to any of ROOT, BRANCH, FILIAL, or REFER to specify the direction of hierarchical expansion, and REPLACE BY HIERARCHY (RH) causes the program to use the new concepts instead of (rather than in addition to) the original ones included in a request.

Phase III currently ends with the hierarchical expansion, and SMART returns control to FMS. One more option, EDITED ANSWERS (EA) follows at this point, producing a final, well edited output page of answers to the search requests. The coding of the option is in process.

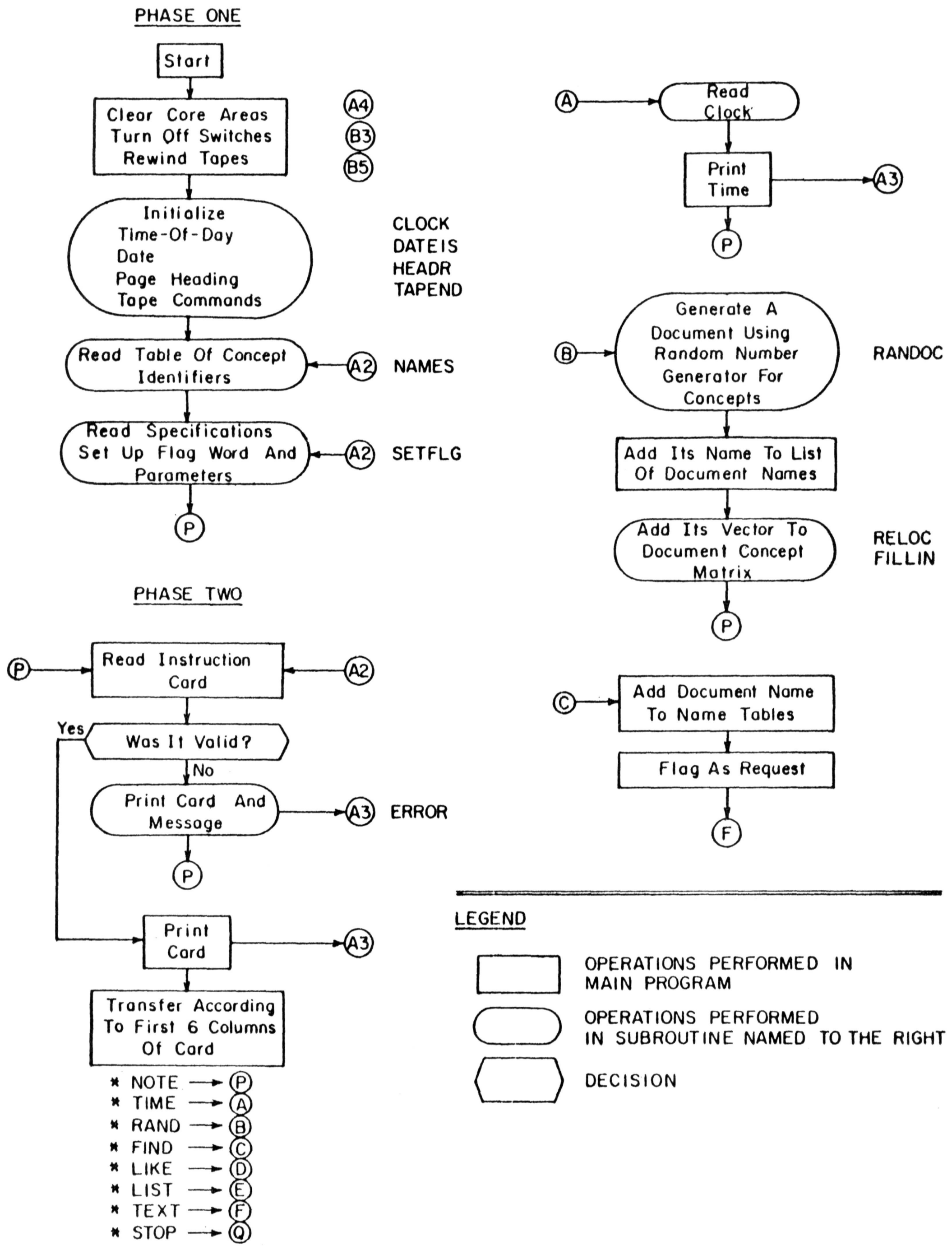
A complete chart of the SMART system flowcontrol, including a specification of the various phases and processing sections is shown in Flowchart 1.

4. Text Read In and Lookup

English text is read into the SMART system by a subroutine named SEGMNT. SEGMNT processes text punched in columns 1-72 of BCD cards; columns 73-80 are ignored. Consecutive blanks are treated as a single blank. A blank is assumed between column 72 of one card and column one of the next card.

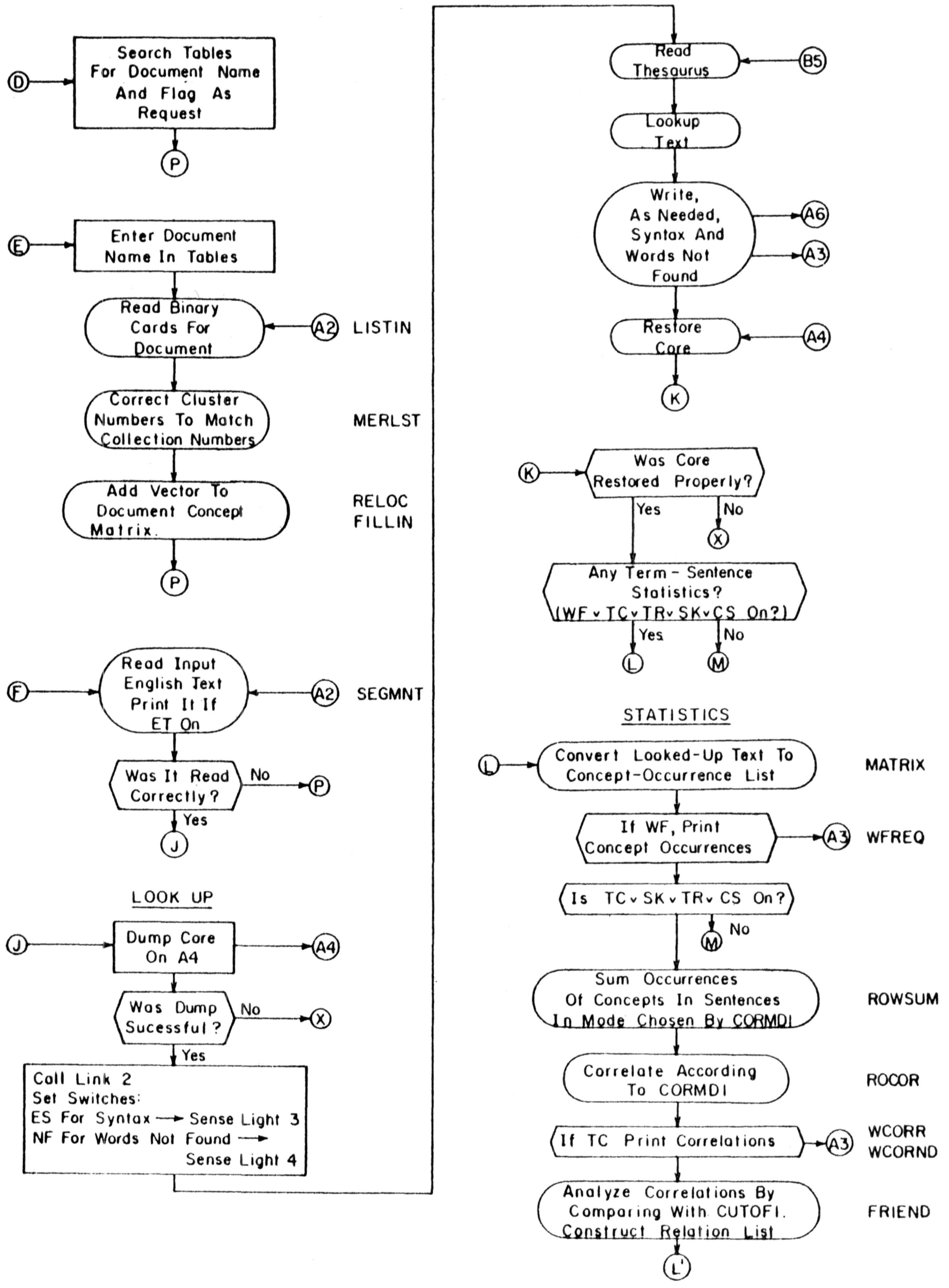
Punctuation marks should be punched as follows:

comma – directly following a given word in a sentence, as is normally done in typing.



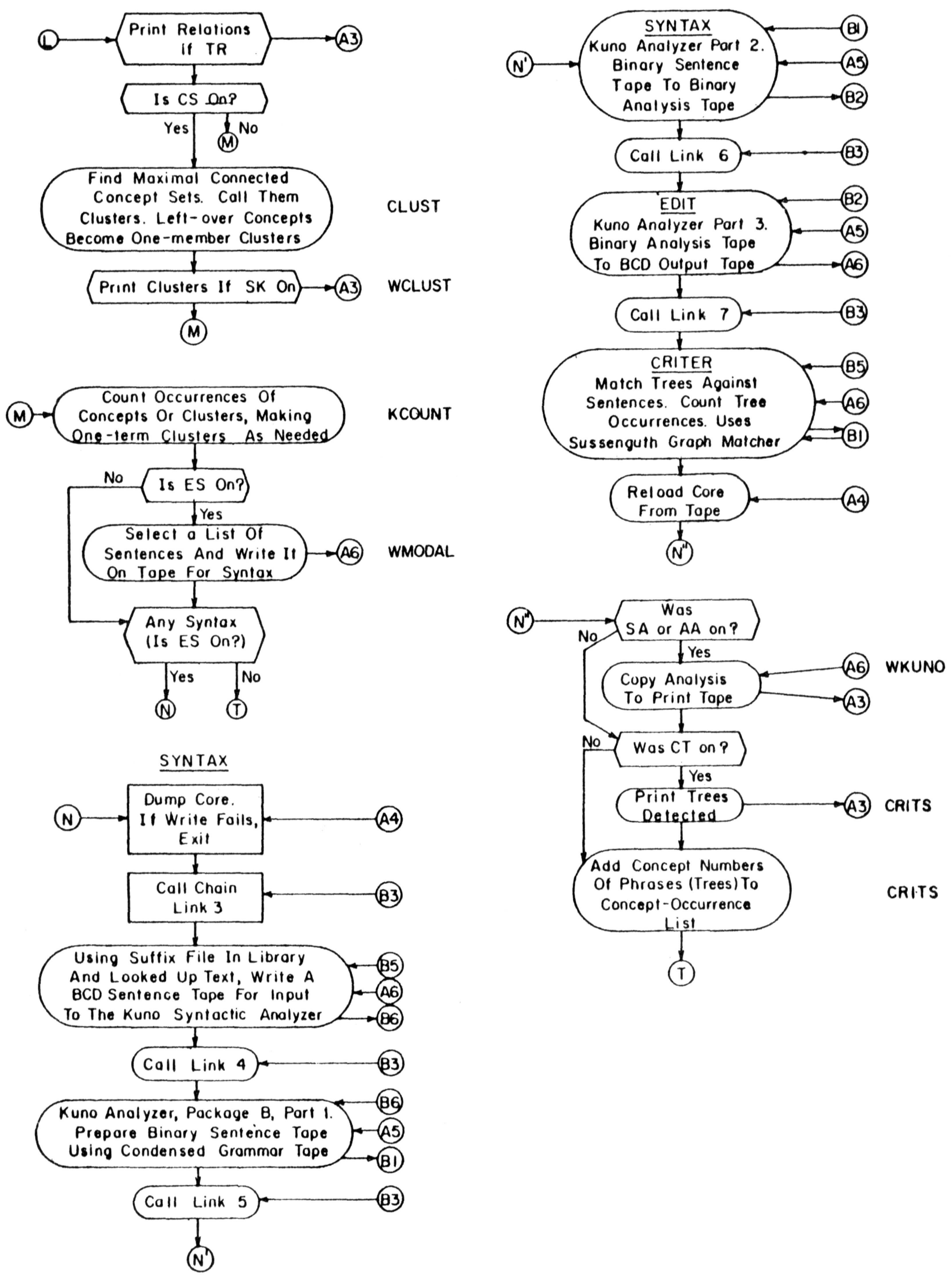
System Supervisor

Flowchart 1



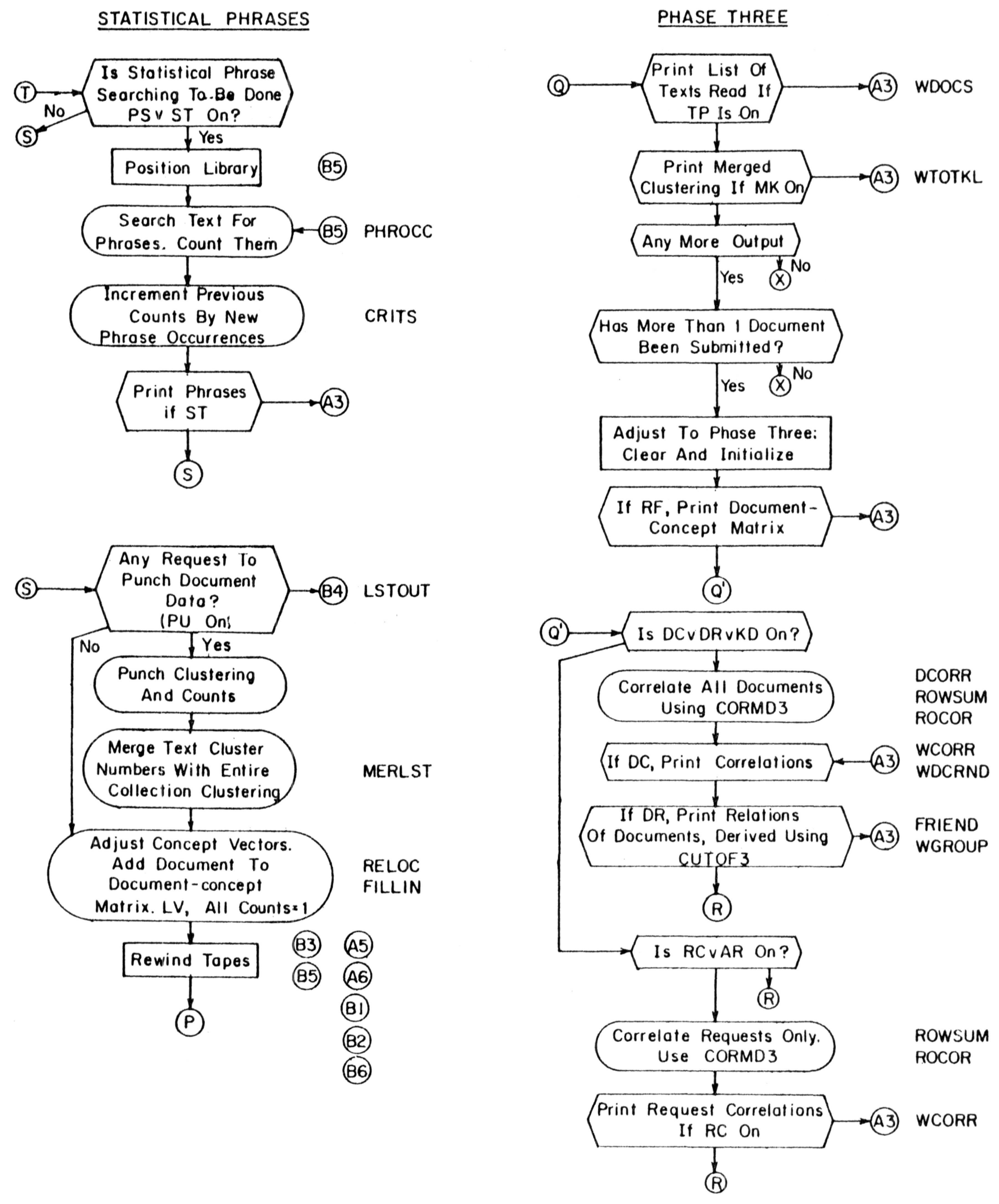
D
Search Tables For Document Name And Flag As Request
P
E
Enter Document Name In Tables
Read Binary Cards For Document
A2
LISTIN
Correct Cluster Numbers To Match Collection Numbers
MERLST
Add Vector To Document Concept Matrix.
RELOC
FILLIN
P
F
Read Input English Text Print It If ET On
A2
SEGMNT
Was It Read Correctly?
No
P
Yes
J
LOOK UP
J
Dump Core On A4
A4
Was Dump Sucessful?
No
X
Yes
Call Link 2
Set Switches:
ES For Syntax → Sense Light 3
NF For Words Not Found → Sense Light 4
Read Thesaurus
B5
Lookup Text
Write, As Needed, Syntax And Words Not Found
A6
A3
Restore Core
A4
K
K
Was Core Restored Properly?
Yes
No
X
Any Term - Sentence Statistics? (WF v TC v TR v SK v CS On?)
Yes
No
L
M
STATISTICS
L
Convert Looked-Up Text To Concept-Occurrence List
MATRIX
If WF, Print Concept Occurrences
A3
WFREQ
Is TC v SK v TR v CS On?
No
M
Sum Occurrences Of Concepts In Sentences In Mode Chosen By CORMDI
ROWSUM
Correlate According To CORMDI
ROCOR
If TC Print Correlations
A3
WCORR
WCORND
Analyze Correlations By Comparing With CUTOFI. Construct Relation List
FRIEND
L'

Flowchart 1 (continued)

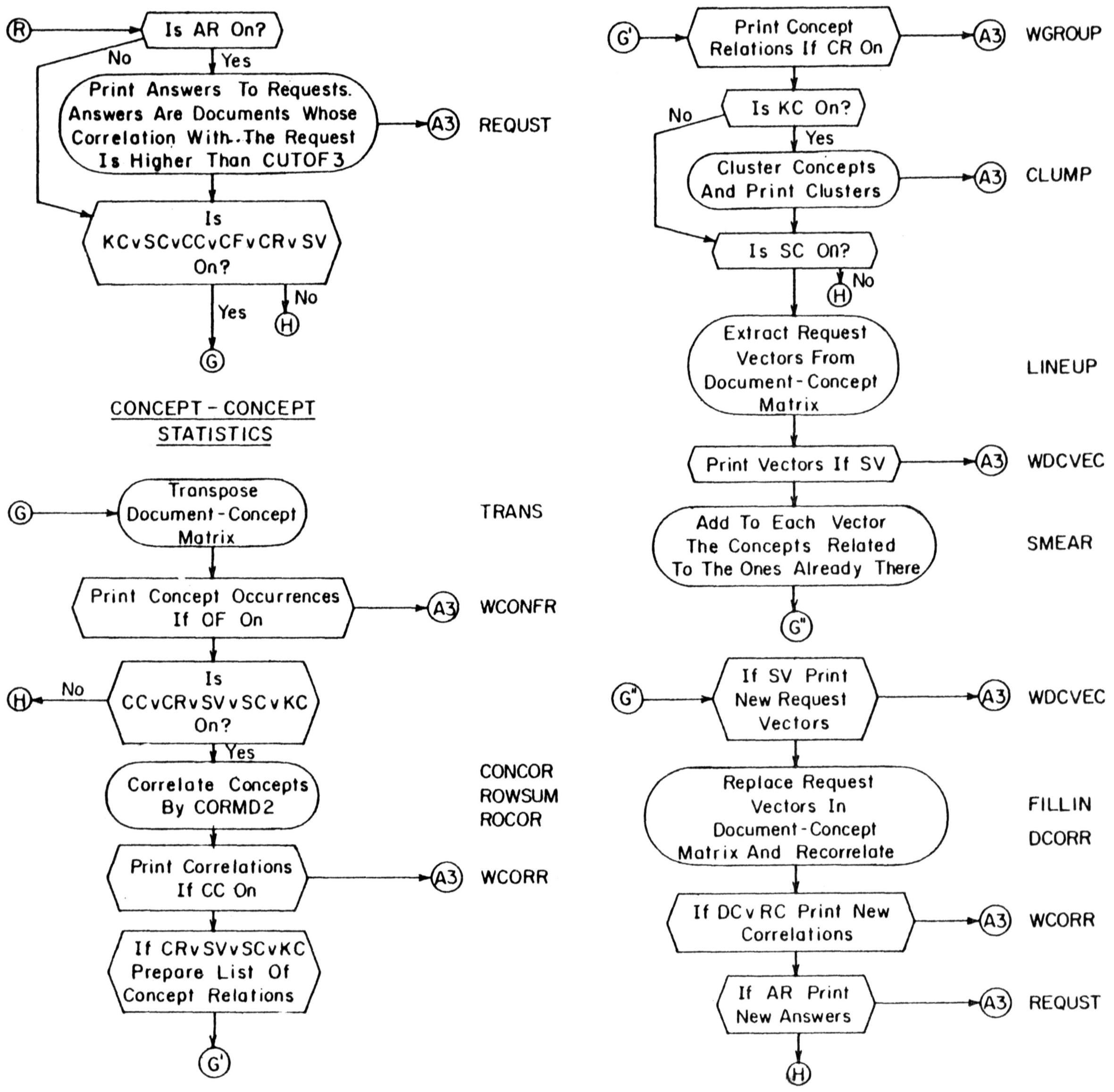


L
Print Relations If TR
A3
Is CS On?
Yes
No
M
Find Maximal Connected Concept Sets. Call Them Clusters. Left-over Concepts Become One-member Clusters
CLUST
Print Clusters If SK On
A3
WCLUST
M
M
Count Occurrences Of Concepts Or Clusters, Making One-term Clusters As Needed
KCOUNT
Is ES On?
No
Yes
Select a List Of Sentences And Write It On Tape For Syntax
A6
WMODAL
Any Syntax (Is ES On?)
Yes
No
N
T
SYNTAX
N
Dump Core. If Write Fails, Exit
A4
Call Chain Link 3
B3
Using Suffix File In Library And Looked Up Text, Write A BCD Sentence Tape For Input To The Kuno Syntactic Analyzer
B5
A6
B6
Call Link 4
B3
Kuno Analyzer, Package B, Part 1. Prepare Binary Sentence Tape Using Condensed Grammar Tape
B6
A5
B1
Call Link 5
B3
N'
N'
SYNTAX Kuno Analyzer Part 2. Binary Sentence Tape To Binary Analysis Tape
B1
A5
B2
Call Link 6
B3
EDIT Kuno Analyzer Part 3. Binary Analysis Tape To BCD Output Tape
B2
A5
A6
Call Link 7
B3
CRITER Match Trees Against Sentences. Count Tree Occurrences. Uses Sussenguth Graph Matcher
B5
A6
B1
Reload Core From Tape
A4
N''
N''
Was SA or AA on?
No
Yes
Copy Analysis To Print Tape
A6
A3
WKUNO
Was CT on?
No
Yes
Print Trees Detected
A3
CRITS
Add Concept Numbers Of Phrases (Trees) To Concept-Occurrence List
CRITS
T

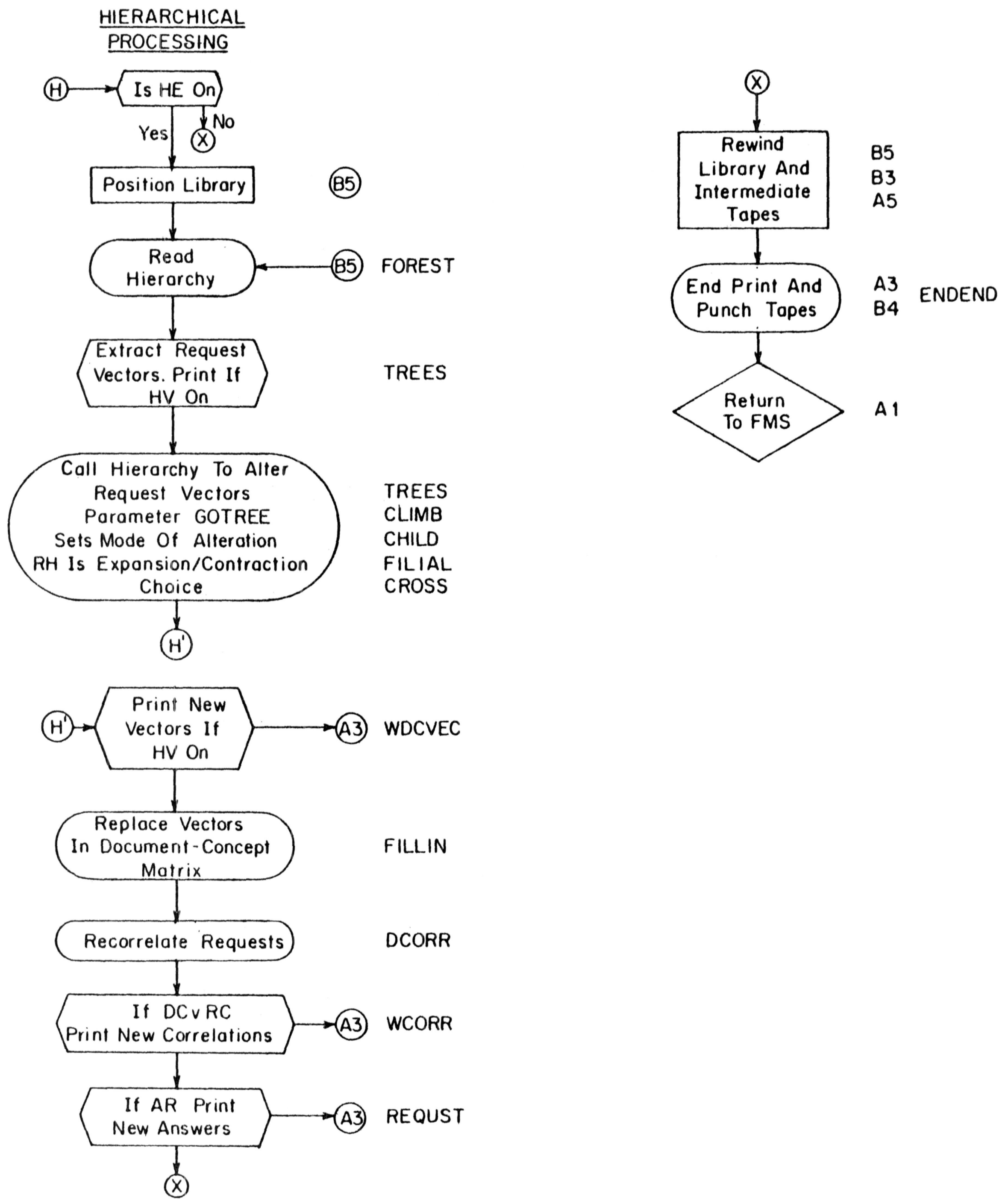
Flowchart 1 (continued)


STATISTICAL PHRASES
T
Is Statistical Phrase Searching To Be Done PS v ST On?
No
S
Yes
Position Library
B5
Search Text For Phrases. Count Them
B5
PHROCC
Increment Previous Counts By New Phrase Occurrences
CRITS
Print Phrases if ST
A3
S
S
Any Request To Punch Document Data? (PU On)
B4
LSTOUT
No
Yes
Punch Clustering And Counts
Merge Text Cluster Numbers With Entire Collection Clustering
MERLST
Adjust Concept Vectors. Add Document To Document-concept Matrix. LV, All Counts=1
RELOC
FILLIN
Rewind Tapes
B3
B5
A5
A6
B1
B2
B6
P
PHASE THREE
Q
Print List Of Texts Read If TP Is On
A3
WDOCS
Print Merged Clustering If MK On
A3
WTOTKL
Any More Output
Yes
No
X
Has More Than 1 Document Been Submitted?
Yes
No
X
Adjust To Phase Three: Clear And Initialize
If RF, Print Document-Concept Matrix
A3
Q'
Q'
Is DCvDRvKD On?
Correlate All Documents Using CORMD3
DCORR
ROWSUM
ROCOR
If DC, Print Correlations
A3
WCORR
WDCRND
If DR, Print Relations Of Documents, Derived Using CUTOF3
A3
FRIEND
WGROUP
R
Is RCvAR On?
R
Correlate Requests Only. Use CORMD3
ROWSUM
ROCOR
Print Request Correlations If RC On
A3
WCORR
R

Flowchart 1 (continued)


R
Is AR On?
No
Yes
Print Answers To Requests. Answers Are Documents Whose Correlation With The Request Is Higher Than CUTOF3
A3
REQUST
Is KCvSCvCCvCFvCRvSV On?
Yes
No
H
G
CONCEPT - CONCEPT STATISTICS
G
Transpose Document-Concept Matrix
TRANS
Print Concept Occurrences If OF On
A3
WCONFR
Is CCvCRvSVvSCvKC On?
No
H
Yes
Correlate Concepts By CORMD2
CONCOR
ROWSUM
ROCOR
Print Correlations If CC On
A3
WCORR
If CRvSVvSCvKC Prepare List Of Concept Relations
G'
G'
Print Concept Relations If CR On
A3
WGROUP
Is KC On?
No
Yes
Cluster Concepts And Print Clusters
A3
CLUMP
Is SC On?
No
H
Extract Request Vectors From Document-Concept Matrix
LINEUP
Print Vectors If SV
A3
WDCVEC
Add To Each Vector The Concepts Related To The Ones Already There
SMEAR
G"
G"
If SV Print New Request Vectors
A3
WDCVEC
Replace Request Vectors In Document-Concept Matrix And Recorrelate
FILLIN
DCORR
If DCvRC Print New Correlations
A3
WCORR
If AR Print New Answers
A3
REQUST
H

Flowchart 1 (continued)


HIERARCHICAL PROCESSING
H
Is HE On
Yes
No
X
Position Library
B5
Read Hierarchy
B5
FOREST
Extract Request Vectors. Print If HV On
TREES
Call Hierarchy To Alter Request Vectors Parameter GOTREE Sets Mode Of Alteration RH Is Expansion/Contraction Choice
TREES
CLIMB
CHILD
FILIAL
CROSS
H'
H'
Print New Vectors If HV On
A3
WDCVEC
Replace Vectors In Document-Concept Matrix
FILLIN
Recorrelate Requests
DCORR
If DCvRC Print New Correlations
A3
WCORR
If AR Print New Answers
A3
REQUST
X
X
Rewind Library And Intermediate Tapes
B5
B3
A5
End Print And Punch Tapes
A3
B4
ENDEND
Return To FMS
A1

Flowchart 1 (continued)

period –	directly after a letter in an abbreviation. End-of-sentence periods should be preceded by a space.
quotes –	slash (/) is used for both open and close quotes. No spaces before close quotes, or after open quotes.
semicolon –	comma period (,.) right after the word.
colon –	two periods (..) preceded by a space (end of sentence implied).
capitalization –	ignore.
question mark or exclamation point –	.QUE or .EP preceded by a space.
dash –	-DASH is used. A minus sign is used for a hyphen. A minus sign followed by a letter is treated as a hyphen (thus: random-access), while a minus sign followed by a blank means skip to the next line.
avoid –	asterisks in column one (this is the normal end-of-text sentinel), and consecutive dollar signs (another end-of-text sentinel).

SEGMNT reads up to 1,000 words of English text. It stores consecutive text words as consecutive five-word items in core, at location WDLIST. The first (FORTRAN-type) word of each item contains the sentence and word numbers: sentence number in the decrement, word number in the address. The remaining four words, in FORTRAN order, contain up to 24 characters of the English word, followed with blank fill. SEGMNT leaves the total number of words in location NUMWRD and the total number of sentences in MAXSEN. If SEGMNT is entered with sense light one on, the text is printed as it is read in.

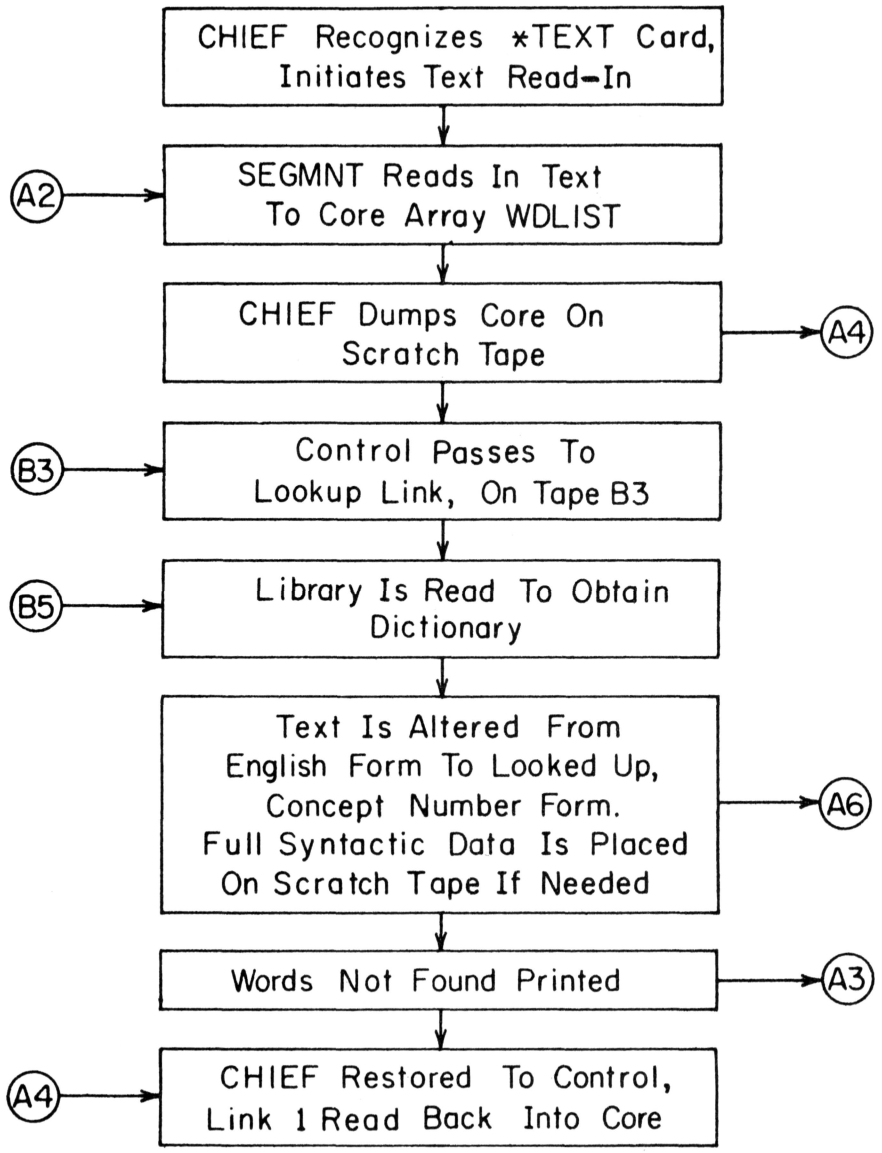
The supervisor now writes all of link one, including accumulated data, onto tape A4. Control is then passed to the dictionary lookup, chain link two. The lookup process is described in Sec. IV in detail. A schematic of the read-in and lookup procedures is shown in Flowchart 2.

5. Operational Details

SMART, as presently coded, requires an IBM 7094 with 12 tapes and 32K of core for full use. If neither dictionary lookups nor hierarchical processing are to be done, the programs will run on an IBM 7090. If syntactic operations are omitted, seven tapes will suffice.

Four of the tapes used by SMART are normal FMS tapes. A1 must contain a FORTRAN monitor system, preferably the Harvard FMS system. This is an almost completely standard FORTRAN II, version 2, modification level 35 system tape. The normal configuration of the monitor file is assumed. The only special feature of the Harvard system is that the job sign-on time is left in 75_8. The date, as usual, is expected in 142_8. During the run the program attempts to read the time from the printer clock (RPQ78C54), expecting a normal SHARE printer board. SMART assumes the IB standard loader. In the absence of the standard loader, the three patches to BSS control made by the SMART system in order to get itself loaded may wreak havoc.

The system input must be on tape A2. Its contents are described later, but its format must be normal; i.e. 14 or 28 word records, <u>with look-ahead coding</u>. SMART reads both binary and BCD data cards. Output


CHIEF Recognizes *TEXT Card, Initiates Text Read-In
A2
SEGMNT Reads In Text To Core Array WDLIST
CHIEF Dumps Core On Scratch Tape
A4
B3
Control Passes To Lookup Link, On Tape B3
B5
Library Is Read To Obtain Dictionary
Text Is Altered From English Form To Looked Up, Concept Number Form. Full Syntactic Data Is Placed On Scratch Tape If Needed
A6
Words Not Found Printed
A3
A4
CHIEF Restored To Control, Link 1 Read Back Into Core

LOOKUP Schematic

Flowchart 2

goes to A3 to be printed under program control. Blocked records, in normal format (lines separated by ≠) are used. Binary cards are written unblocked on B4, in order to be punched.

SMART also requires a scratch tape on A4 to be used by the main link and also as a chain tape. The main chain tape, however, is B3. Links 2 to 7 are stacked on it; only links 11, 10, and 1 go on to A4. Normally, when no updating is done, link 1 is submitted as a nonchain job and a previously written chain tape is mounted on B3.

The SMART library tape goes on B5. If a new library tape is to be written in this run, B5 should be blank and the old library tape must be mounted on A6. Otherwise, the library is mounted directly on B5 and updating is omitted.

If syntactic processing is performed, five more tapes are needed. The analyzer itself is on B3 with the other chain links. The SMART version of the syntactic analysis differs from the standard version by

(1) having different tape assignments;

(2) running as three chain links rather than as three jobs;

(3) printing fewer on-line messages;

(4) producing (optionally) only one analysis of each sentence.

The condensed grammar is mounted on A5. Grammar 6C is used currently in the SMART system. The syntactic system also uses A6, B6, B1, and B2 as intermediate tapes. A full SMART run (with syntax but no update) thus requires three private tapes (B3, B5, A5), four monitor tapes (A1, A2, A3, B4) and five scratch tapes (B1, B2, A4, A6, B6).

Having mounted all these tapes, further operator intervention is not required. All further instructions are taken from A2.

The format of the input tape is as follows:

(1) Job card.

(2) *XEQ.

(3) Operation instructions, as needed.

(4) Programs.

```
                     ┌ * CHAIN (11,4)
                     │   binary deck for chain 11
    if updating      ┤
                     │ * CHAIN (10,4)
                     └   binary deck for chain 10

    if more than     { * CHAIN (1,4)
    one chain link

                         binary deck for link 1

                     ┌ * CHAIN (2,3)
    if chain tape    │       :
    not premounted   ┤       :
                     └ * CHAIN (7,3)
```

(5) *DATA

(6) Optional updating instructions, used if and only if links 11 and 10 are submitted.
Submit:
   (a) lookup file update cards
   (b) suffix file update cards
   (c) criterion tree update cards
   (d) statistical phrase update cards
   (e) hierarchy update cards

(7) Table of names of thesaurus categories. This is a table of six-character identifiers used for printouts. There is one identifier included for each concept number.

Three formats are allowed for the names table.

(a) BCD format. One card is given for each identifier. These cards have *NAMES punched in columns 1-6, and the six-character identifier in columns 7-12. The concept number may be punched as a decimal integer, beginning in column 13 and continuing for up to five columns. Each concept number must be less than 1,000 in magnitude. If no concept number is provided (i.e. column 13 is blank), the concept number used for the preceding card is incremented by one. When all needed identifiers have been supplied, a card with ****** in columns 1-6 terminates the deck.

(b) Binary formed. When a correct BCD deck is submitted, a binary deck is punched. This deck may be used as future input. The first card is a BCD card containing an octal integer in columns 1-3, specifying the number of cards that follow. The remainder of the deck is binary; it has 4-7-9 punches in column one, and contains 25 identifiers per card.

(c) Omit names table entirely. The program will supply consecutive integers.

(8) Specifications: At this point the options described in Part 3A, and listed in Table 1, must be supplied. All specifications for CHIEF are punched, in any order, on BCD cards. Any number of cards and any positions on the cards may be used. Blanks are ignored and commas used to separate specifications. Since all printing is under the control of some option, at least one output-producing option must be specified for each

run. Read in of the options stops when the first card of type nine is recognized.

(9) Instructions to CHIEF: From this point on, the input deck consists of the following kinds of cards:

(a) *NOTE

(b) *TIME

(c) *LIKE

(d) *LIST, followed by a binary deck

(e) *TEXT, followed by a BCD text

(f) *FIND, followed by a BCD text (do not use)

(g) *STOP. Last card of deck

(10) End of file: Superfluous if a *STOP card is included, but desirable nonetheless.