VI. THE DICTIONARY SETUP PROCEDURES

M. Cane

1. Introduction

The present section describes the routines used to set up the thesaurus (or concept dictionary). The specifications for the input deck are explained in detail, including also a discussion of all setup options. The setup section creates the thesaurus and suffix trees used by the thesaurus lookup programs. These trees are then stored as the first file of a library tape.

2. Input Deck, Control Card, and Data Card Formats

Inputs to the setup programs consist of a single control card specifying what is to be done, followed by additional cards containing the data for the entries to the suffix tree (if any), and the thesaurus tree (if any).

The control card is divided into two fields: columns 1-6 and columns 7-12. All fields are left-justified. Table 1 lists the possible configurations of these fields and the options they control.

First Field col. 1-6	Second Field col. 7-12	OPTION
BOTH		Both the suffix and thesaurus trees are set up from cards. Nothing is done with the old tape.

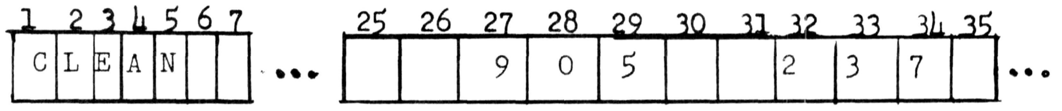
First Field col. 1-6	Second Field col. 7-12	OPTION
BOTH	SPACE	Same as above, except that the file of the old library tape with the lookup data is spaced over.
COPY		All trees are copied from the old library tape – no further input cards are expected.
THES		The suffix tree is copied from the old library; the thesaurus is set-up from input cards.
SUFFIX		The suffix tree is set up from input cards; the thesaurus is copied from the old library tape.

Thesaurus and Suffix Setup Control Card Formats
Table 1

The suffix data cards (if any) follow the control card. Each card contains two fields: The first (columns 1-12) contains the English suffix; the second (columns 13-15) contains a number less than $2^8 = 256$ associated with each suffix in a one-one correspondence. The contents of the first field must be left-justified (i.e., the first letter of the suffix must occur in column 1), and the contents of the second field must be right-justified (e.g., a number such as 97 consisting of two digits, would appear as a "9" in column 14, and the "7" in column 15; column 13 would be zero or blank). The suffix data thus corresponds to a FORTRAN format of (2A6, I3).

The first suffix enteredmust begin with the letter "e" (c.f., the description of subroutine LOOK in report ISR-7, IV-18). The termination of the suffix data is signaled by a card with the letter "Z" in columns 1-6.

The thesaurus data (if any) appear last. Each thesaurus input card is divided into 15 fields. The first (columns 1-24) contains the English word to be entered, left-justified within the field. The next six fields (columns 25-54) contain the semantic codes (category numbers) associated with the word. Each code is allotted five columns, so that the codes appear in columns 25-29, 30-34, 35-39, 40-44, 45-49, and 50-54. Each code must be right-justified within its allotted field. Records must also be entered consecutively, starting with the left-most semantic field. For example, if there are two codes, say 905 and 237, associated with the word "CLEAN," these must appear in the first two fields. The card would then appear as follows:



The category numbers must be less than $2^{15} = 32768$ in magnitude.

The last eight fields (columns 55-80) contain the syntactic codes associated with the given word. These codes must again be left-justified within the entire syntactic field and must be entered consecutively. Each code must be right-justified within its field. The syntactic codes must be less than $2^{8} = 256$ in magnitude. The thesaurus cards may therefore be read with a format of 4A6, 6I5, 8I3. The input words must be in strict binary coded decimal (BCD) order. The end of the thesaurus data is signaled by a card with 0-8-5 punches in column 1, and *END* in columns 2-6.

3. Implementation of Setup

The subsection explains how the setup process is accomplished. A general discussion is presented first, followed by a description of the component programs.

First the suffix tree[†] is created, either from the input data cards, or by copying the old library tape; the suffix tree is then written onto the new tape. The old library tape (if any) should be mounted as tape A6, and the new tape as B5. The thesaurus tree is then created (or copied) and written onto the new tape.

In order to allow a thesaurus of unlimited length, the thesaurus is divided into blocks by the setup programs. Each block approximately fills the space allotted to the thesaurus in the lookup link. All words beginning with thé same letter must fit within one block (see description of TIPUT, below). A block may contain words with different initial letters. All words beginning with a given letter are read in as a unit. They are added to the block now being created if this can be done without exceeding the storage allotted to this block. Otherwise, the present block is written onto the tape, and a new block is started. This process is repeated until

[†] For a general discussion of tree structures and their use in the SMART lookup schemes see Report No. ISR-7, Section IV.

all the input cards are exhausted. A word is associated with each block gives the highest (relative to the BCD ordering) initial letter of any which word in the block (c.f., description of subroutine LOOK below).

The following programs comprise the dictionary setup routine:

(1) SWRITE and SREAD respectively write and read the suffix tree. They have one argument, specifying the top location of the tree.

(2) TWRITE and TREAD, respectively, write and read a block of the thesaurus. They each have two arguments: The first specifies the top location of the thesaurus block; the second is the highest initial letter of the block.

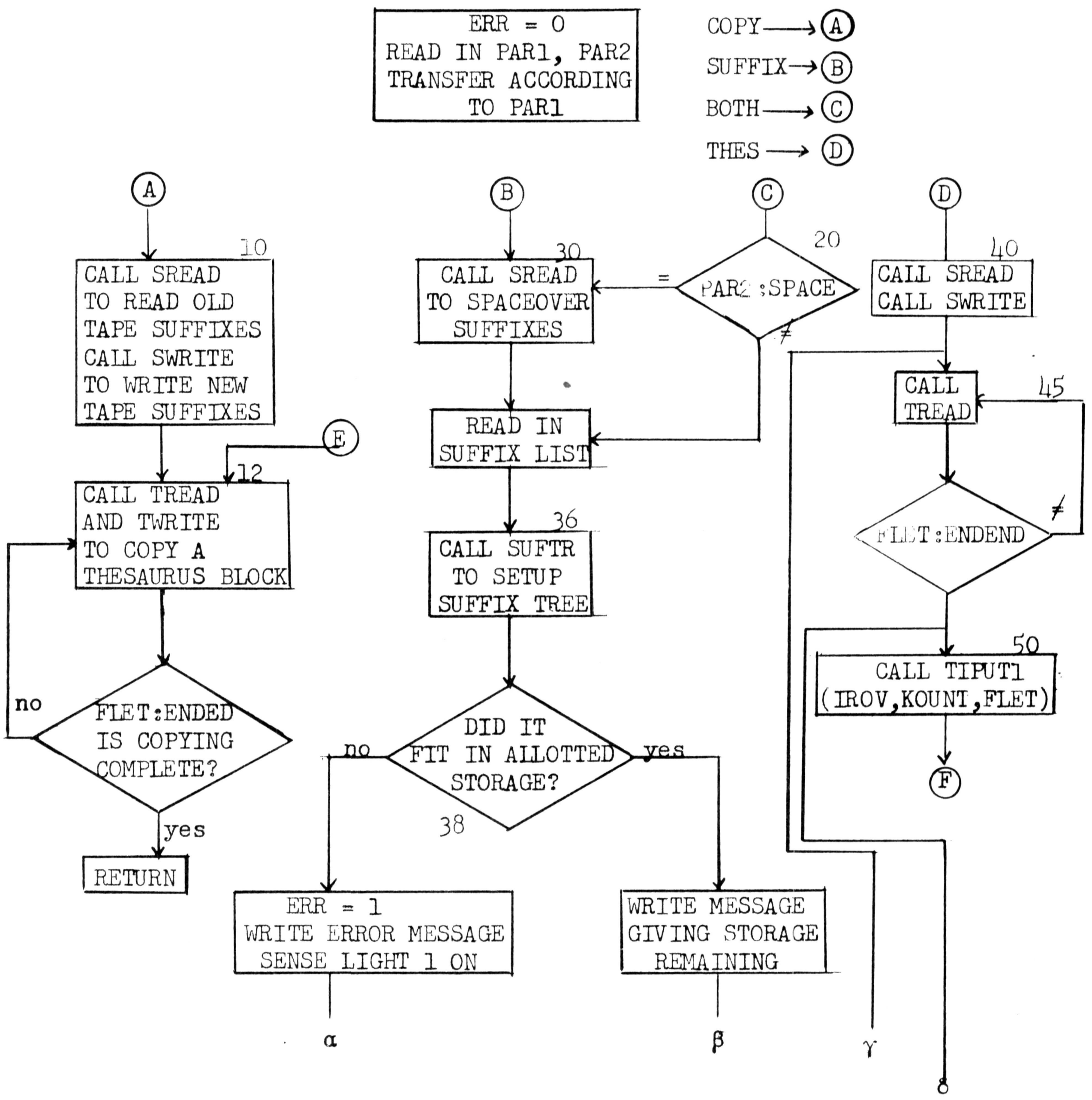
(3) SUFTR sets up the suffix tree. It is described in detail in report ISR-7, IV-4. SUFTR has four arguments: The first two indicate where the input starts and how many items are included. The last two arguments are returned by SUFTR. The first of these gives the top location of the tree. The last, if positive, indicates the number of unused locations allotted to the tree. If the last argument has a negative sign, it gives the number of locations by which the tree exceeded its allotted storage.

(4) TREET and TRADD set up the thesaurus tree. The former is an entry for starting a new tree (or a new block), and the latter adds to an existing tree. They each have three arguments, corresponding to the last three arguments of SUFTR. These are described in detail in report ISR-7, IV-4.

(5) TIPUT and TIPUT1 read in the thesaurus data and convert the semantic and syntactic codes to the proper internal format. The semantic codes **are** packed two to a word in decrement and address respectively. The syntactic codes are packed into eight 8-bit fields, with the fifth field split between two words. The last eight bits of the second word are left empty.

TIPUT1 is used as initial entry to TIPUT. Both programs have three arguments, all of which are created by the programs themselves. The first is a flag which is zero for a normal return, negative if the program returns because its buffer is filled, and positive if the card signalling the end of the thesaurus inputs has just been read. The second is the counts of words put into the buffer, and the third indicates the initial letter of the words in the buffer. TIPUT reads-in all and only those words beginning with the same letter before returning control (except for the case where its buffer is filled before this is accomplished, corresponding to a negative first argument).

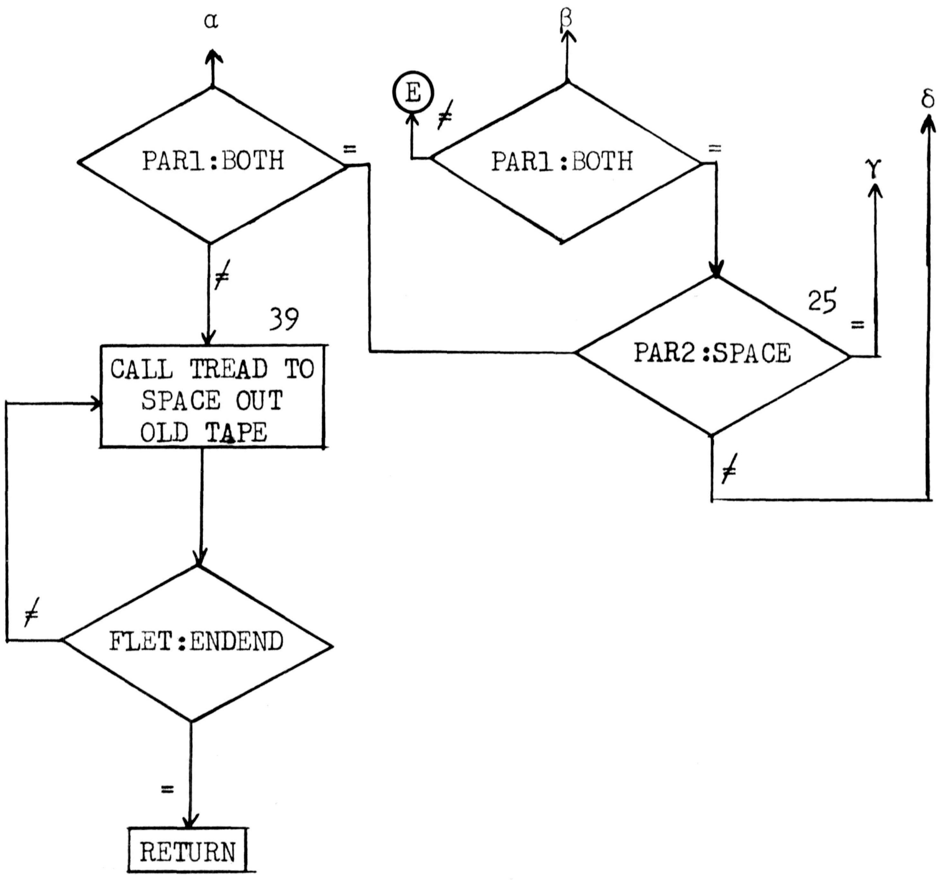
TIPUT also checks to see if the word just read-in begins with a letter different from the initial letter of the previous word. To do this, a word is set up whose first 6-bits contain the last initial letter found and whose remaining bits are all "on." The initial machine word of the next English word is compared against this word; if the former is greater, a new initial letter is present; if not, the present letter is taken to be the same as the previous letter. This procedure implies that the deck need not be in strict BCD order; in particular, it will suffice for the cards to be in order with respect to only column 1.



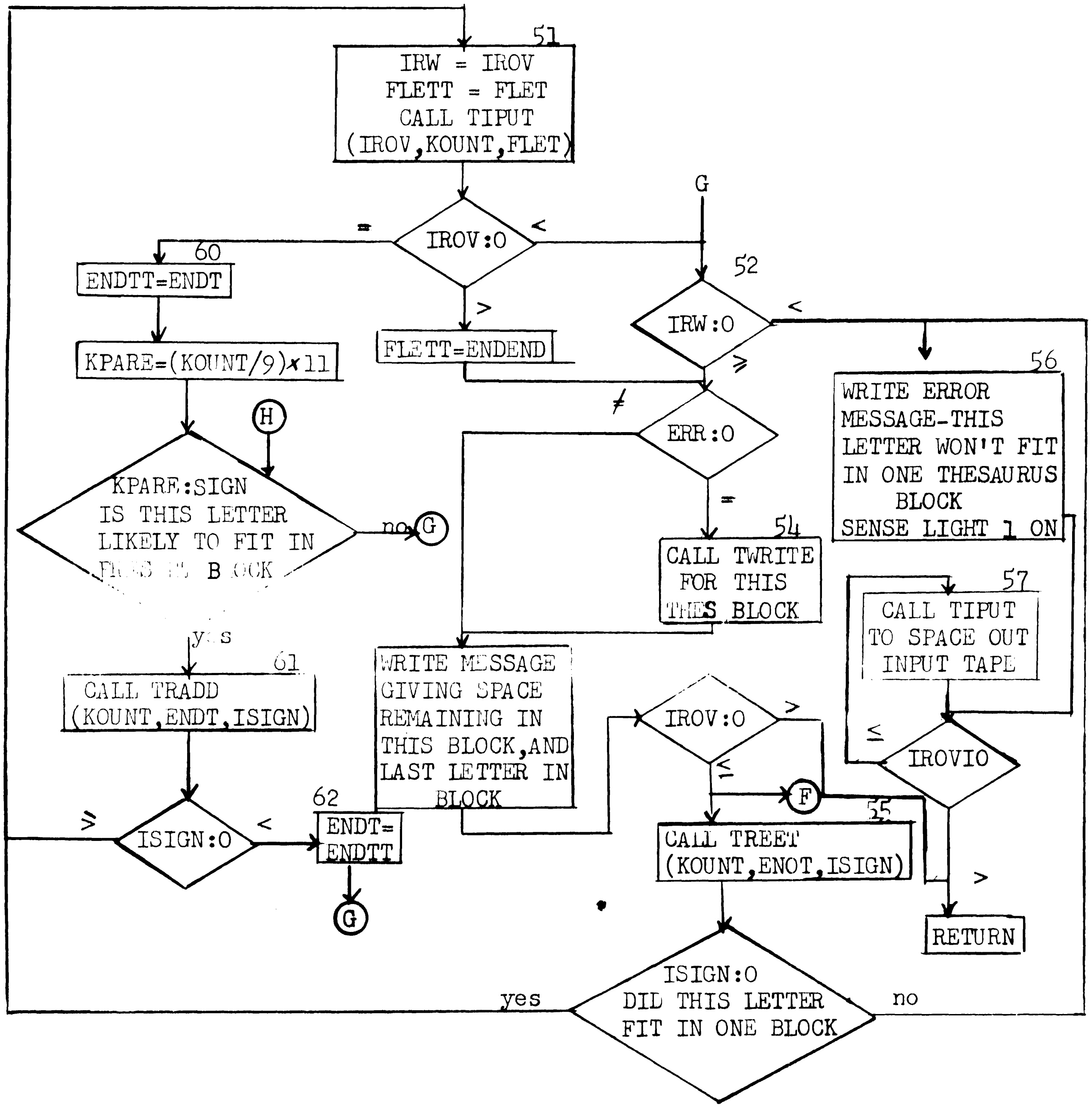
ERR = 0
READ IN PAR1, PAR2
TRANSFER ACCORDING TO PAR1
COPY → A
SUFFIX → B
BOTH → C
THES → D
A
10
CALL SREAD TO READ OLD TAPE SUFFIXES CALL SWRITE TO WRITE NEW TAPE SUFFIXES
E
12
CALL TREAD AND TWRITE TO COPY A THESAURUS BLOCK
FLET:ENDED IS COPYING COMPLETE?
no
yes
RETURN
B
30
CALL SREAD TO SPACEOVER SUFFIXES
READ IN SUFFIX LIST
36
CALL SUFTR TO SETUP SUFFIX TREE
DID IT FIT IN ALLOTTED STORAGE?
38
no
yes
ERR = 1 WRITE ERROR MESSAGE SENSE LIGHT 1 ON
WRITE MESSAGE GIVING STORAGE REMAINING
α
β
C
20
PAR2:SPACE
=
≠
D
40
CALL SREAD CALL SWRITE
45
CALL TREAD
FLET:ENDEND
≠
50
CALL TIPUT1 (IROV,KOUNT,FLET)
F
γ
δ

Subroutine CANE - Supervisor For The
Dictionary - Thesaurus Setup

Flowchart 1



Flowchart 1 (continued)


51
IRW = IROV
FLETT = FLET
CALL TIPUT
(IROV,KOUNT,FLET)
G
IROV:0
=
<
>
60
ENDTT=ENDT
KPARE=(KOUNT/9)x11
FLETT=ENDEND
52
IRW:0
<
≥
56
WRITE ERROR MESSAGE-THIS LETTER WON'T FIT IN ONE THESAURUS BLOCK
SENSE LIGHT 1 ON
H
KPARE:SIGN
IS THIS LETTER LIKELY TO FIT IN [illegible] BLOCK
no
G
yes
ERR:0
≠
=
54
CALL TWRITE FOR THIS THES BLOCK
57
CALL TIPUT TO SPACE OUT INPUT TAPE
61
CALL TRADD (KOUNT,ENDT,ISIGN)
WRITE MESSAGE GIVING SPACE REMAINING IN THIS BLOCK,AND LAST LETTER IN BLOCK
IROV:0
>
≤
F
IROVIO
≤
>
55
CALL TREET (KOUNT,ENOT,ISIGN)
62
≥
ISIGN:0
<
ENDT=ENDTT
G
RETURN
ISIGN:0
DID THIS LETTER FIT IN ONE BLOCK
yes
no

Flowchart 1 (continued)

If the dictionary were to become so large that all words beginning with a given letter could not fit in one block, it would be necessary to base a division into blocks on more than just the initial letter. This can be accomplished by changing the parameter MASK in subroutine TIPUT to ensure that the first two letters, for example, be taken into account. All further tests (i.e., those in LOOK) work properly, no matter what the block subdivision turns out to be.

(6) CANE is the supervisory program for the dictionary setup. It is the only program called from the main program. Flowchart 1 describes the details of this program; a few additional remarks may be needed. The numbers appearing above the upper right-hand corner of some of the boxes key the flowchart to the statement numbers of the program. In case of error, sense-light 1 is turned on to inform the main program. The test in the box labelled "H" seeks to determine whether the words just read-in can be added to the block without overfilling it. The test is based on the empirical estimate that an item takes an average of eleven locations when converted to tree form. If this test indicates that these words probably will not fit in the block, control is transferred to the box labelled "G," to initiate a sequence which writes out the present block and begins a new block with these new input words.

If the test succeeds, control passes to the block labelled "61," which adds the new words to the present block of the tree. A new test is then performed to make sure that the allotted storage has not been exceeded. If it has, the end of the block is reset to the previous point (box 62), and control is transferred to the sequence which writes out the block.