VOLUME II | NR. 3 | OCTOBER 1969

STUDIES

MOUTON

COMPUTER STUDIES

in the Humanities and Verbal Behavior

BOARD OF EDITORS

Anthropology	PAUL KAY, University of California, Berkeley
Archaeology	GEORGE L. COWGILL, Brandeis University
Art	CHARLES CSURI, The Ohio State University
	LESLIE MEZEI, University of Toronto
Bibliography	ERIC BOEHM, American Bibliographical Center, Santa Barbara
Classics	LEONARD BRANDWOOD, The University of Manchester
	T. M. ROBINSON, University of Toronto
Education	ELLIS B. PAGE, University of Connecticut
Folklore	JOHN Q. ANDERSON, University of Houston
Graphics and Algorithms	JEFFREY RASKIN, University of California, San Diego
History	THEODORE K. RABB, Princeton University
Library Science	RALPH H. PARKER, University of Missouri
Linguistics	SHELDON KLEIN, University of Wisconsin
	ROBERT S. WACHAL, University of Iowa
Literature and Modern Languages	BERTRAND AUGST, University of California
	RICHARD BAILEY, University of Michigan
	JESS BESSINGER, JR., New York University
	WILLIAM INGRAM, University of Michigan
	HENRY KUČERA, Brown University
	JAMES W. MARCHAND, Cornell University
	STEPHEN PARRISH, Cornell University
	ALICE POLLIN, New York University
Mass Communication	WILLIAM J. PAISLEY, Stanford University
Mathematics	PAUL R. HALMOS, Indiana University
Music	LEJAREN HILLER, SUNY, Buffalo
Philosophy	LARRY TRAVIS, University of Wisconsin
Political Science	OLE R. HOLSTI, University of British Columbia
Programming	DANIEL E. BAILEY, University of Colorado
Psycholinguistics	JOHN B. CARROLL, Educational Testing Service, Princeton
Social Psychology	HANS E. LEE, Michigan State University
Sociology	WALTER A. SEDELOW, University of Kansas
Speech	EDWARD E. DAVID, Bell Telephone Laboratories
	GERALD M. SIEGEL, University of Minnesota
Statistics	JULIET SHAFFER, University of Kansas
	HAROLD P. EDMUNDSON, University of Maryland
Theater	GARY GAISER, Indiana University
Translation	SILVIO CECCATO, Università di Milano
	DAVID A. DINEEN, University of Kansas

EDITORIAL ADDRESS

Computer Studies in the Humanities and Verbal Behavior is sponsored by the University of Kansas. Manuscripts: Sally Y. Sedelow, Department of Computer Science, University of Kansas, Lawrence, Kansas 66044, U.S.A. General correspondence: FLOYD R. HOROWITZ, Department of English, University of Kansas, Lawrence, Kansas, U.S.A.

Computer Studies in the Humanities and Verbal Behavior appears quarterly in issues of approximately sixty pages. Four issues constitute a volume. The subscription price is $ 10.00/*f* 36.00 per year per volume; single issues cost $ 3.00/*f* 10.50. Back issues as well as subscriptions and single issues can be ordered from every bookseller or subscription agency, or directly from Mouton & Co., P.O. Box 1132. The Hague, The Netherlands.

A Computer Graphics System for Non-Alphabetic Orthographies*

S. DUNCAN, T. MUKAII, and S. KUNO
Harvard University, Cambridge, Massachusetts

This paper describes new developments on a system, originally developed at Harvard and since expanded to include other non-standard orthographies, which uses a PDP-1 computer with cathode-ray tube display and a tablet as input devices, and a Stromberg-Carlson's 4020 recorder as an output device. With the use of this system, a text may be encoded in any orthography or combination of orthographies. Now included in the system's repertoire are: 8,000 Chinese characters, all Japanese syllabic characters, all Korean syllabic characters, the Tamil syllabic characters, the Pahlavic, Hebrew, and English alphabets, and a set of mathematical and other special characters. New characters may be added as they are encountered during encoding by drawing the character on a 16 × 16 scope grid. In its present form the system is accessible only to those at Harvard or those with identical hardware. Portions of the system, however, have been used by several other institutions engaged in computer research with non-standard (especially Chinese) characters.

1. INTRODUCTION

This paper describes new developments on a system which was originally designed as an answer to the need for inputting and outputting Chinese texts for automatic processing.[1] A detailed description of the organization of the original system may be found in Hayashi, Duncan and Kuno (1968). The system, developed at Harvard, has since been expanded to include other non-standard orthographies as well. The system uses a PDP-1 computer with cathode-ray tube display scopes and a tablet as input devices, and a Stromberg-Carlson's 4020 recorder as an output device. Each character is internally represented by pairs of co-ordinates. Each pair corresponds to the beginning and end points of each of the straight lines which constitute the character on the 16 × 16 grid which is standard for all of the characters in the repertoire of the system.

With the use of this system, a text may be encoded in any orthography or combinations thereof. The repertoire of the system currently includes 8,000 Chinese characters, all Japanese syllabic characters, all Korean syllabic characters, the Tamil syllabic characters, the Pahlavic, Hebrew, and English alphabets, and a set of mathematical and other special characters. New characters may be added to the system's repertoire as they are encountered during the encoding of a text. This is accomplished by drawing the character on a 16 × 16 grid displayed on a scope.

The encoding of characters in a text may be accomplished by selecting with a special Rand pen, the text characters from character tables which have been prepared for the Rand tablet. The running encoded text may be edited both before and after it has been stored on magnetic tape. Upon input and output, characters are always manipulated only in their natural orthographic form, eliminating the need for manually encoding and decoding arbitrarily assigned codes. The text inputting component of this system may be likened to a typewriter with no mechanical parts; the encoding of a text proceeds as quickly as the user can find the text characters in characters tables and as quickly as he is able to select them with a pen. The process is similar to typing with one finger on a typewriter keyboard of some 3,000 keys. For an experienced user, this process can be quite rapid.

* Research reported here has been supported in part by the National Science Foundation under Grant No. GN-554 and GS-1934, by the American Mathematical Society under a subcontract to Harvard from the National Science Foundation's Grant GN-644 to the AMS, by the Advanced Research Projects Agency under Contract SD-265 to Harvard and by the Air Force under Contract No. F-19628-68-C-0379. — A preliminary version of this paper was presented at the Symposium for Computers in Chinese Language, Translation and Exotic Orthographies held at Arlington, Virginia on April 11, 1969, under the sponsorship of Technical Operations, Inc.

[1] See: G. L. Walker, S. Kuno, S. N. Smith, and R. B. Holt, "Chinese mathematical text analysis", *IEEE Transactions on Engineering Writing and Speech*, Vol. EWS-11, No. 2 (August 1968), pp. 118-128; S. Hayashi, S. Duncan, and S. Kuno, "Graphical input/output of non-standard characters", *The Communications of the ACM*, Vol. 11, No. 9 (September 1968), pp. 613-618.

The system in its present form is accessible only to those at Harvard or equipped with an identical hardware configuration. However, it has already been possible to extract portions of the system for use at several other institutions engaged in computer research with non-standard, particularly Chinese, characters.

2. VARIOUS TECHNIQUES FOR ENCODING CHINESE CHARACTERS[2]

Unlike the English alphabet, which is a small, finite set of characters that can be combined according to certain rules to form meaningful units of words, the Chinese character set is virtually unlimited. This is particularly evident in dealing with ancient Chinese texts or in dealing with documents which contain proper nouns. Since the number of characters in common use is very large, although much smaller than the possible set (which is about 50,000 characters), it is impracticable to construct any sort of mechanical recording device which retains a one-to-one correspondence between characters and 'keys', and which guarantees that all the characters that one may encounter in processing Chinese texts of various types may be found on the keyboard. Thus, it has heretofore been necessary to employ some sort of alphanumeric coding system in order to prepare Chinese texts for computer manipulation.

The peculiarities of Chinese characters just discussed call for a method which relies on the principle that the most readily comprehensible means of encoding is one which retains their natural graphical shape in visible form. Before discussing our system, we shall present the advantages and disadvantages of several techniques which have been used to encode Chinese texts. The need for an approach of the type we have chosen will become apparent.

2.1 *Manual Encoding*

Chinese characters can be encoded in a way that reflects their sound value, with or without tone markings; that is, by pronunciation codes. However, more than one character may be represented by a single pronuciation, introducing ambiguities into the coded text which were not present in the original. Furthermore, the sound values of Chinese characters vary from dialect to dialect, and thus, a text may be encoded quite correctly in more than one way. And, while the original text is intelligible to all literate speakers of Chinese, regardless of dialect, a text encoded with the sound values of one particular dialect may be completely unintelligible to a speaker of another dialect.

Chinese characters may be encoded in a way that reflects their physical configurations. The four-corner system is an example of this approach. This is a classification system based on the configurations of each Chinese character at its four corners. Each character is represented by a five-digit sequence which does not have to be remembered for each character, but can be derived by the configuration of the character once the system is mastered. However, because of the similarities of the strokes at the four corners of some characters, the system is ambiguous in that more than one Chinese character can be assigned the same four-corner system index. Moreover, a single character may be encoded in more than one way, and the encoded text is not readily intelligible to native speakers or scholars of Chinese. A refinement of the system devised in Russia makes it possible to uniquely encode otherwise ambiguous characters by assigning additional digits, but has the disadvantage that these unambiguous codes are no longer derivable from simply examining the configurations of the characters, and that they are hard to remember.

Chinese characters may be encoded by some arbitrary, pre-determined code which bears no relation to the character's sound value or physical configuration. From the point of view of unambiguous coding, this approach is efficient. The system which is most generally recognized and used is the telegraph code,[3] which represents each character by a sequence of four digits. Although a text encoded by telecode is unambiguous, since it is simply a sequence of unique four-digit numbers, it is totally unintelligible to all but professional telegraphers, and therefore cumbersome to code and decode.

2.2 *Encoding Machines*

Chinese characters may be encoded by such devices as the Japanese teleprinter or the Chinese Monotype. Both of these devices are similar to a Flexowriter in that the typewriter keyboard is co-ordinated with a paper tape perforator. Each character is uniquely represented by a certain series of perforations on paper tape. These devices have the advantage that Chinese characters can be input graphically by depressing the keys which contain the imprint of these characters. One can view these devices as typewriters with 2,400 to 4,600 character keys.[4]

[2] A more comprehensive survey of input/output techniques and devices for Chinese characters is given in Kierman and Barber (1968).

[3] See Dougherthy, Lamb and Martin (1963).

[4] Ordinary Japanese/Chinese teleprinters have a capacity of some

These devices have the drawbacks that if a character which is not on the keyboard is encountered, there is no way of encoding it uniquely and obtaining a printout of the character, that a bulky off-line printer is required, and that a text may not be proof-read and edited during encoding. In addition to the engineering difficulties mentioned above, other drawbacks of the teleprinter are that it is not easy to add new characters to the system, that the text may not be proofread and edited during encoding, and that the system is not amenable to other non-standard orthographies, such as mathematical symbols, which have totally different internal structures, if any, of strokes and curves.

The Ideographic Composing Machine (ICM) of RCA[5] is also based on the physical configurations of the components of Chinese characters. The keyboard of the ICM contains keys for the basic strokes which occur in the drawing of characters. A character is encoded by depressing keys for strokes in the same sequence as used in writing. As soon as a given sequence unambiguously identifies a character by a process of matching against the 10,000 stored character configurations,[6] the image of the character is displayed for operator inspection. If the operator finds the character to be correct, he depresses the photographic key and the character is imprinted on film. When the sequence of key depressions for an entire character still does not identify the character unambiguously, all the characters with the same sequence are displayed on the scope and the operator may select the correct one.

There are several mechanical devices which rely on an encoding of Chinese characters based, not on their entire configurations, but on the components of the characters. The Chicoder, developed by the Itek Corporation, is an example of such a device. The Chinese characters are encoded by depressing two keys representing the upper half or left-side configuration and the lower half or right-side configuration of the characters. The depression of two keys causes all the Chinese characters with the same specifications to be displayed on the scope, and the user chooses the one that he wants to encode by depressing keys specifying the location of that character on the scope. Each character is represented by a unique 18-bit code on the perforated paper tape. An off-line printer has been added to the Chicoder for printing Chinese characters thus encoded. With respect to the current state of the Chicoder, the following quotation from Kierman and Barber's survey is informative:

> To our knowledge, only a single version of the Chicoder was ever produced; and this was delivered to the Air Force in 1966 and is now being used for experimental purposes at the University of Texas. Originally conceived as an input device only, the Chicoder has never been satisfactory in the output mode which Itek tried to build in: discrepancies occur, and about 40% of the output characters are simply unreadable.[7]

Although the ICM seems to be the best encoding and printing device thus far discussed, it does have the following drawbacks. First, it can only be used by those who know the correct sequence of strokes for all the Chinese characters in its repertoire. Also, there are some characters which have more than one stroke configuration. The user must know exactly which configuration has been adopted by the system for each such character. Secondly, the ICM does not have the capability of adding new characters to the system in the course of encoding a text or editing the text on-line. Thirdly, it is an expensive special-purpose machine costing somewhere around $100,000.

Of somewhat unique nature is the experimental Kanji display system (Hayashibara and Saito, 1968) developed by Oki Electric Industry Co., Ltd., Japan.[8] This system, like the teleprinters that were discussed previously, is based on the total configurations of Chinese characters. The current version contains 2,500 characters, with each key containing four characters. The encoded characters are output on perforated paper tape for possible off-line or on-line computer manipulation. What is unique about this system is that it has a cathode-ray tube for displaying up to 400 of the characters thus far encoded. Characters are displayed on the scope by a dot method; each character is represented by the presence or absence of dots at each intersecting point on an 18 × 18 matrix. The system also has a facility for editing the encoded text, including correction, erasure, insertion as well as rearrangement. The system cannot at present produce hard copy. A version which has a repertoire of more than 5,000 characters is being designed. This machine, when coupled with a printing device, seems to be one of the most promising as an office typewriter for the Japanese and Chinese orthographies; but one distinct drawback to the system is its limitation on the allowable character set.

2,400 characters. Each key contains four characters, and therefore the keyboard consists of some 600 rectangular keys. The encoding of a character is accomplished by depressing the key which contains that character and selecting a shift-key corresponding to the position of the character on the key. The Chinese teletypewriter (East West Computer Corporation), designed by C. C. Kao, has a capacity of some 4,600 characters.

[5] F. E. Shashoua, *Photocomposition machine for the Chinese language* (Camden, New Jersey, Applied Research, Defense Electronic Products, Radio Corporation of America, 1964).

[6] Chinese characters are stored on microfilm in the graphical form. Therefore, the style of the characters of the ICM is extremely elegant.

[7] Kierman and Barber (1968), p. 39.

[8] H. Hayashibara and K. Saito, "Kanji display system", *Japan Electronic Engineering*, September 1968, pp. 30-36.

From the above discussion of approaches to encoding Chinese characters, it must be clear that a system is called for (1) which has the feature of permanent storage of new characters as they are encountered in the course of text encoding, (2) which eliminates the formidable requirement that the operator have knowledge of special codes or conventions regarding Chinese characters, and (3) which permits the proofreading and editing of a text as it is encoded. The graphics system currently in operation at Harvard fills these very needs.

3. HARVARD GRAPHICS SYSTEM FOR NON-STANDARD CHARACTERS

Our graphics system consists of three components:

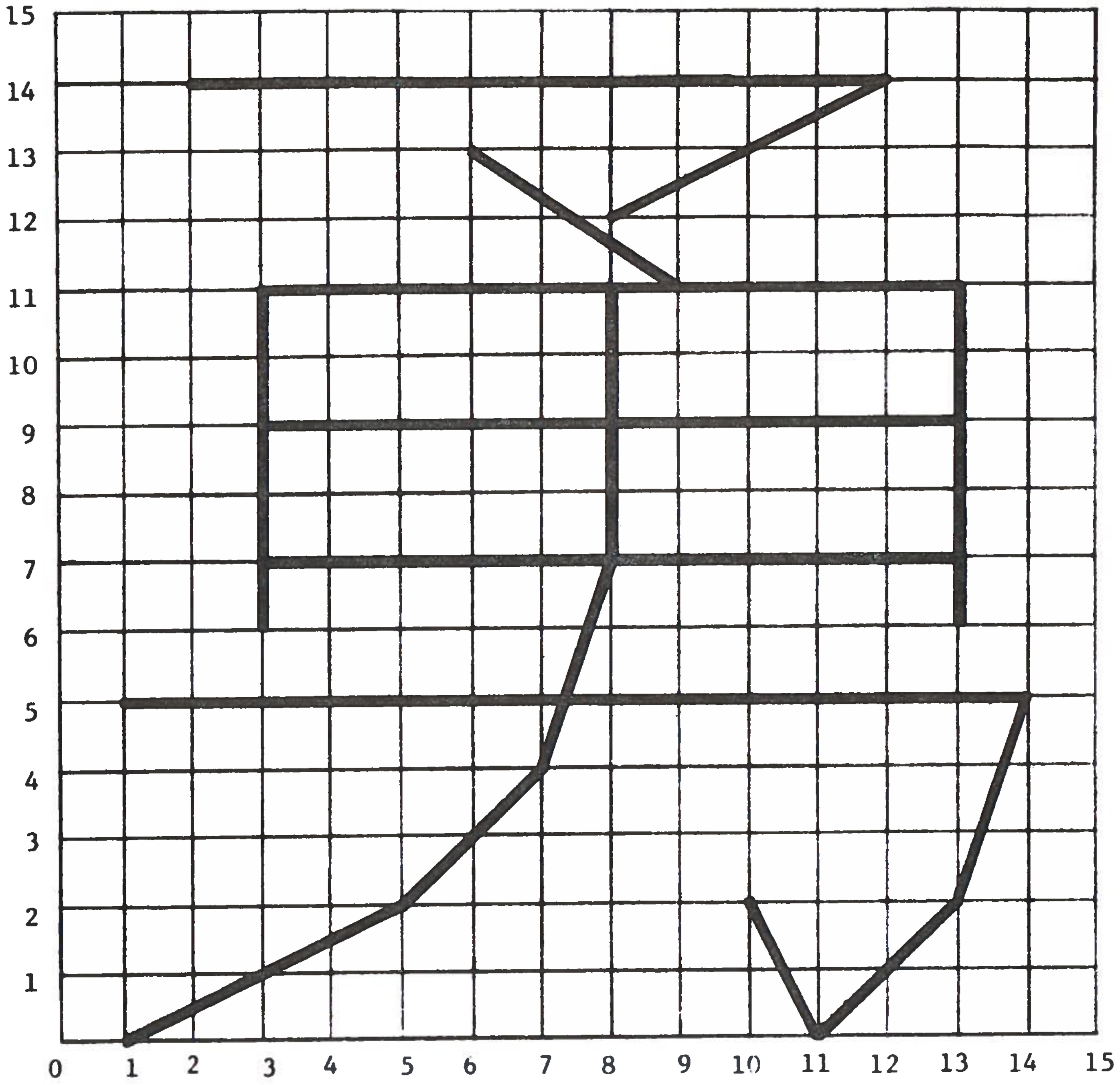


Fig. 1. A character drawn on a 16 × 16 grid.

(1) Text-Encoding: a text is encoded and edited in the orthography of the text language

(2) Character Entry: new characters are added to the repertoire of the system either off-line or during the text-encoding process

(3) Text Printing: a text is printed in its own orthography.

The first two components are on the PDP-1 with the aid of the attached scopes and Rand Tablet, whereas the Text Printing component operates on a S-C 4020 recorder.

3.1 *Character Representation*

All characters in the repertoire of the system are represented by sequences of straight lines on a 16 × 16 grid. For example, the Chinese character for 'brave' can be written on a 16 × 16 grid with 16 straight lines as shown in Figure 1. The configuration of this character is internally represented by pairs of X and Y co-ordinates of these 16 straight lines (read from left to right):

(2,14) (12,14); (12,14) (8,12); (6,13) (9,11);
(3,11) (3,6); (3,11) (13,11); (13,11) (13,6);
(3,9) (13,9); (3,7) (13,7); (8,11) (8,7);
(1,5) (14,5); (14,5) (13,2); (13,2) (11,0);
(11,0) (10,2); (8,7) (7,4); (7,4) (5,2); (5,2) (1,0)

The topmost horizontal line of the character is internally represented by the co-ordinate pair (2,14) (12,14) designating the beginning point and the end point of the line on the grid. The beginning and end points of each straight line comprising the character must correspond to a grid point. A curved line which starts at the center of the character and terminates at the left bottom corner is drawn as a sequence of three short straight lines, which are represented internally by the three pairs of co-ordinates (8,7) (7,4); (7,4) (5,2); and (5,2) (1,0).

Since the system is not dependent upon the peculiar stroke system of the Chinese characters, it is amenable to any orthography as long as its characters can be represented by sequences of straight lines on a 16 × 16 grid. Each character thus represented is assigned a unique alphanumeric code (four digit code in the case of Chinese characters).

At present, the repertoire of the system consists of the Chinese, Japanese, Korean, English, Tamil, Hebrew, Pahlavic, and mathematical orthographies. The constraint that all characters must be represented by straight lines on a 16 × 16 grid has not caused any difficulties in plotting any characters thus far entered into the system. Orthographies consisting of curly lines, such as Burmese scripts, are expected to be difficult to implement on the system, and elegance of style might have to be sacrificed for the convenience of this grid system.

3.2 *Text Encoding*

In the original experimental version of the system which was described in Hayashi, Duncan and Kuno (1968), text characters to be encoded were selected with a light-pen from character tables displayed on the scope. Since only 50 characters can appear on the scope simultaneously[9] the system's 4,000 Chinese characters were divided into many small tables, based on their radicals. Each table had to be accessed individually with great cost in time.

In the current system, characters are selected from 10 × 10 inch printed tables prepared for placement on the Rand tablet. Each Rand tablet table has a capacity of up to 3,024 characters, as may be seen in Figure 2. Thus, the repertoire of Chinese characters may be divided into only a few such character tables and the rather lengthy time spent in specifying a desired scope character table and waiting for it to appear on the scope is eliminated.

Rand tablet character tables are produced by an S-C 4020. Therefore, it is not necessary to manually write character tables each time they undergo changes in terms of character arrangement. All that is needed to produce new character tables is to specify the new order of characters using the characters' unique alphanumeric codes, and the S-C 4020 will produce printouts of the new tables.

In encoding a text, the user places an appropriate character table on the Rand tablet, be it a character table for the Japanese orthography, the Chinese orthography, the mathematical symbols, or the Tamil orthography. The user then points with the Rand pen to the table identification,[10] and then to the character he wishes to encode. The table identification and the location of the character within the table which have been indicated are translated into a unique position in a special internal table which contains a pointer for each character to the computer location of the sequence of pairs of co-ordinates representing that character. For example, the character for 'brave' appears as the 11th character of the 9th line of Character Table 1. If this table is placed on the Rand tablet, and if this character is selected by the user with the Rand pen, the location of this character in Table 1 is translated into internal

[9] Because of the resolution on a CRT display scope and the problem of flicker, readability of characters would decrease drastically if more characters were to be placed on the scope at a time.

[10] In Figure 2, the table ID is '1' as may be seen in the upper left portion of the table.

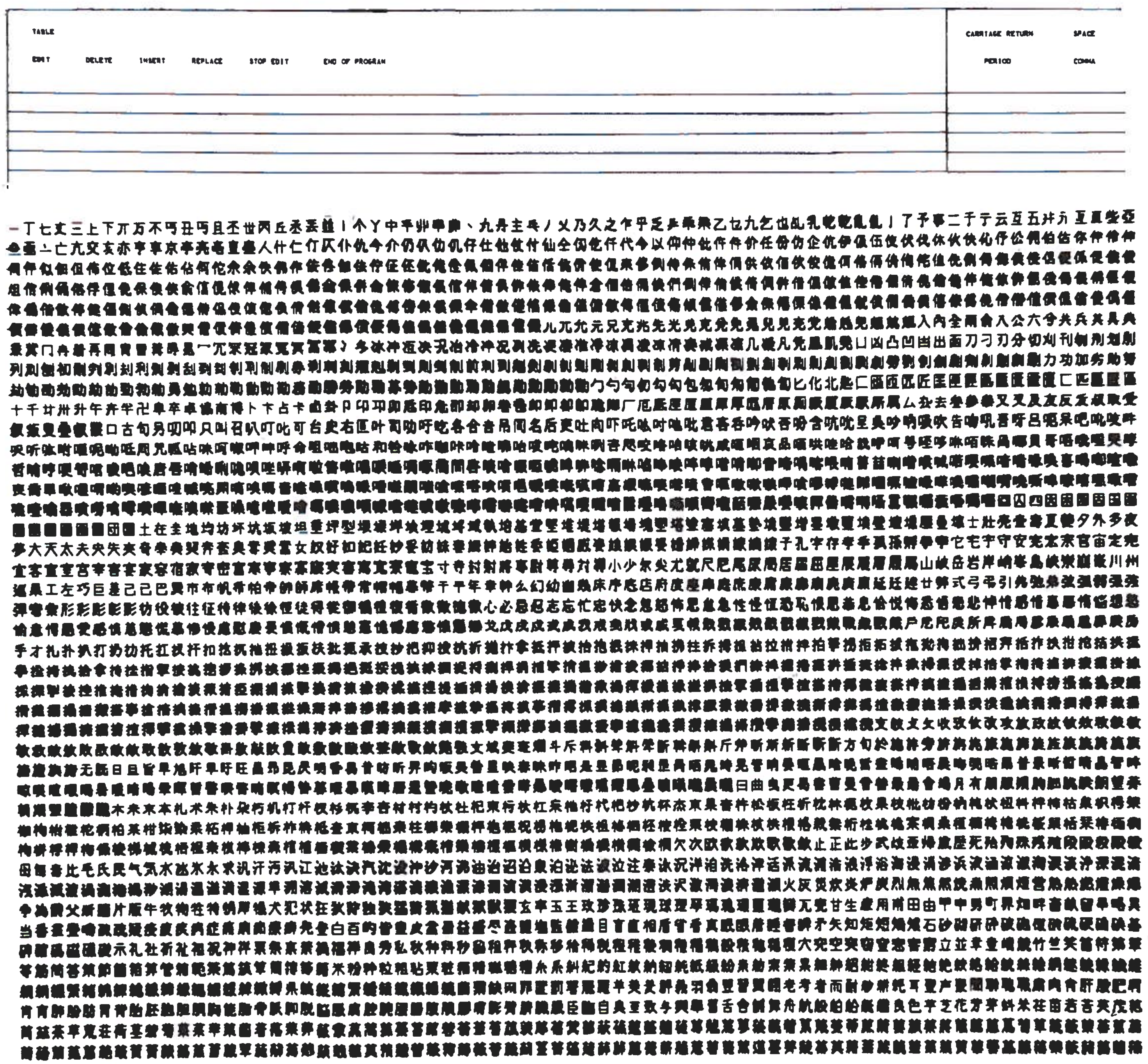


Fig. 2. The first Chinese table for the Rand tablet.

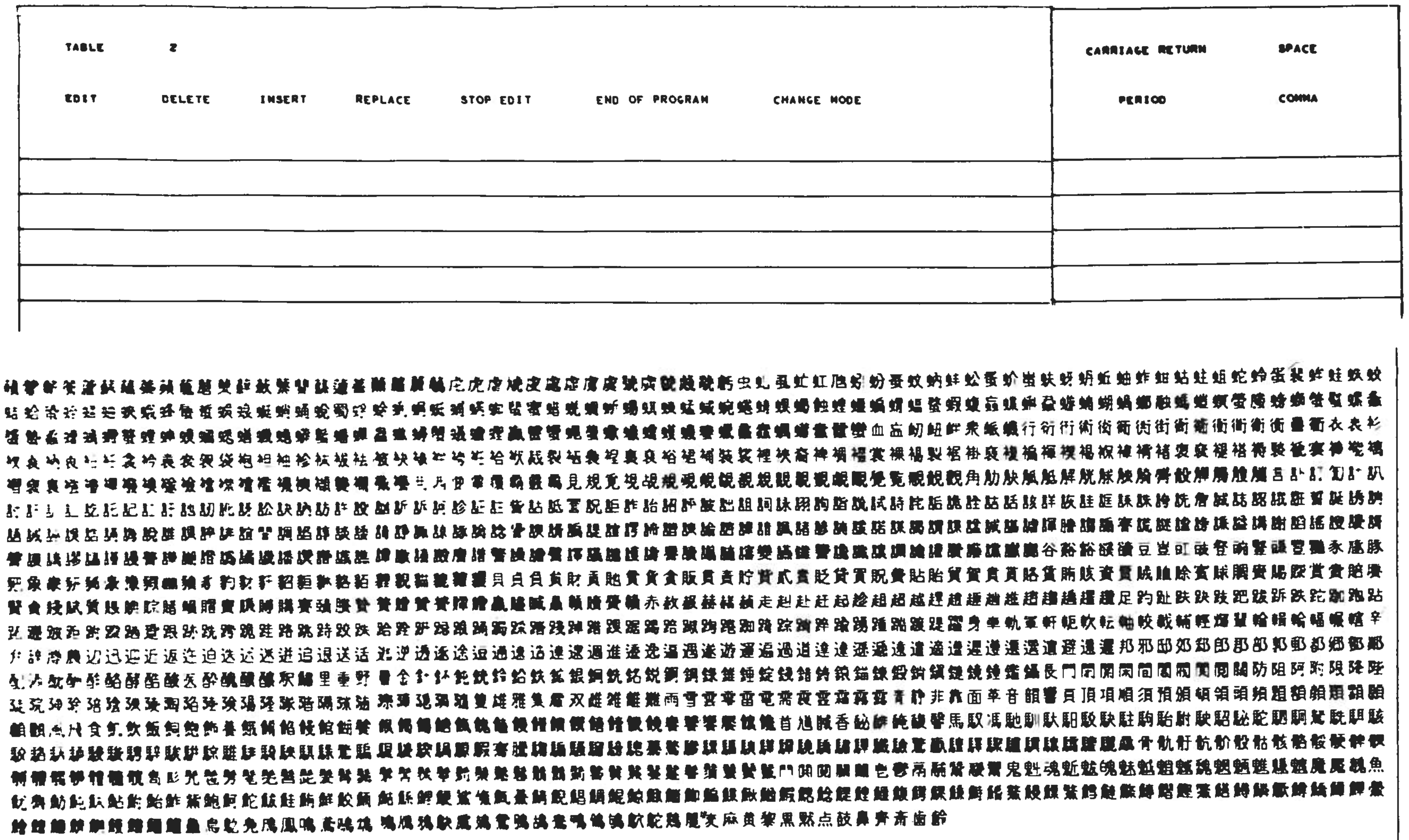


TABLE 2

EDIT DELETE INSERT REPLACE STOP EDIT END OF PROGRAM CHANGE MODE

CARRIAGE RETURN SPACE

PERIOD COMMA

[illegible]

Fig. 3. The second Chinese Rand table.

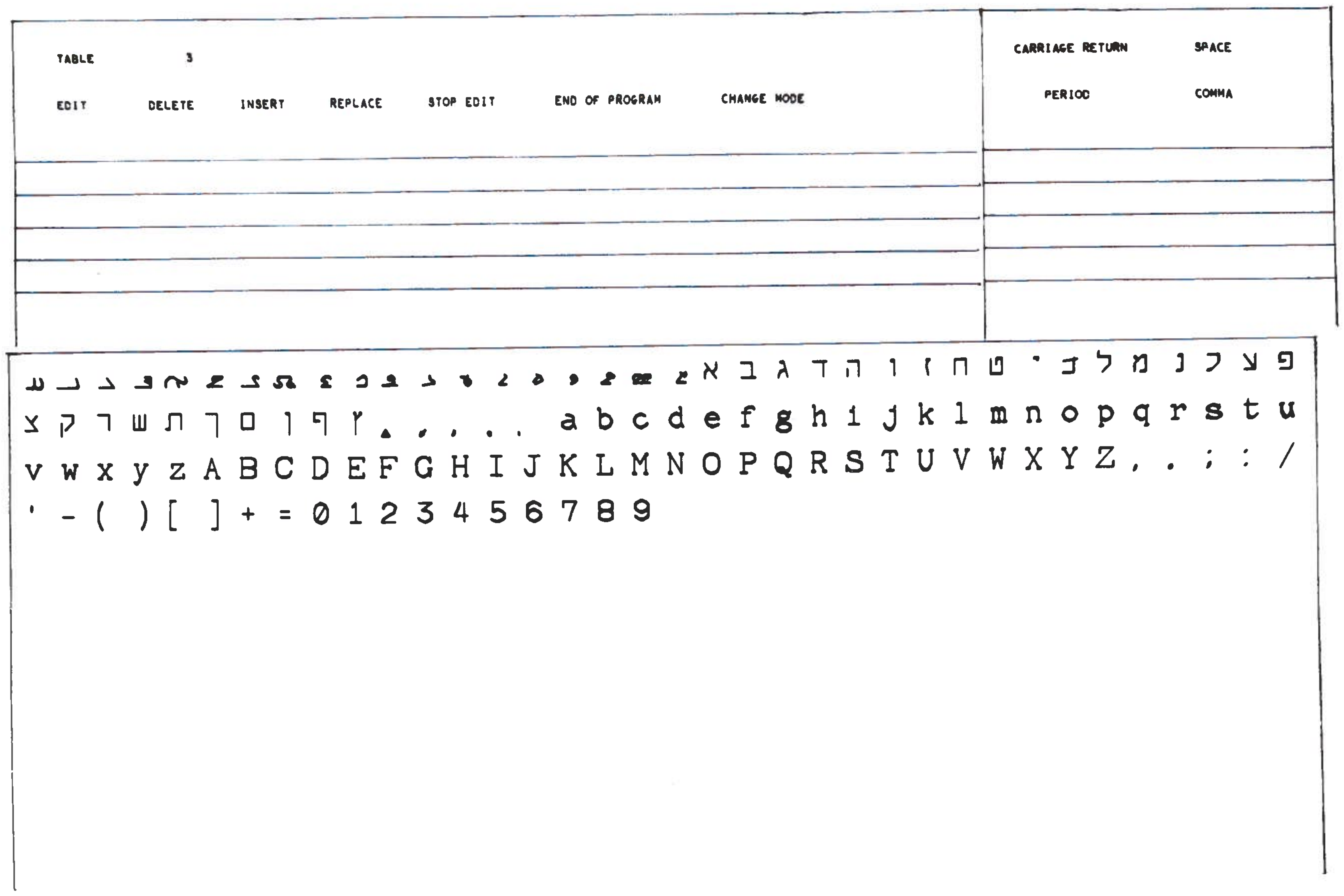


TABLE 3
EDIT
DELETE
INSERT
REPLACE
STOP EDIT
END OF PROGRAM
CHANGE MODE
CARRIAGE RETURN
SPACE
PERIOD
COMMA
a b c d e f g h i j k l m n o p q r s t u
v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z , . ; : /
' - ( ) [ ] + = 0 1 2 3 4 5 6 7 8 9

Fig. 4. The Rand table of Pahlavic and Hebrew characters.

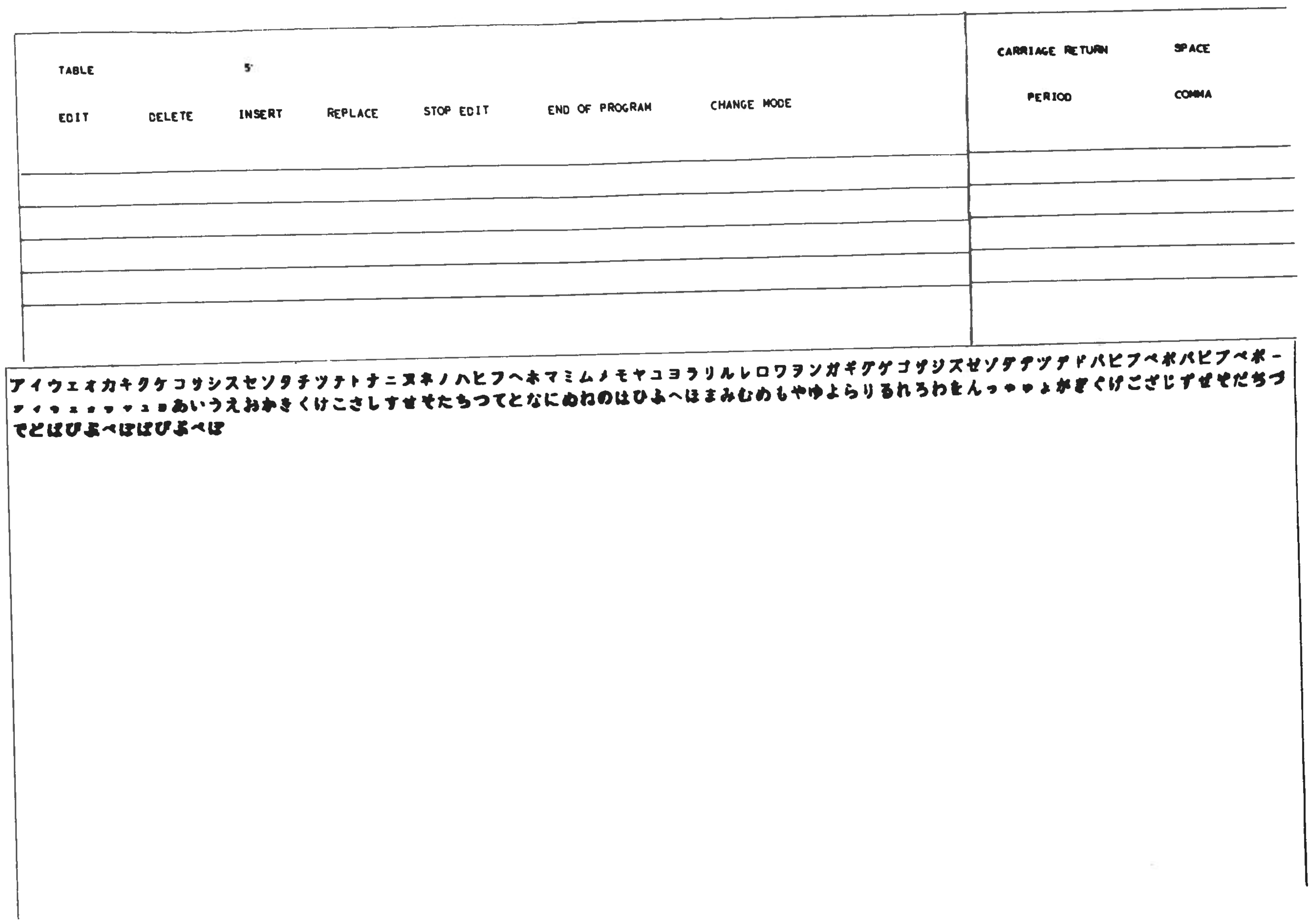


TABLE
5
EDIT
DELETE
INSERT
REPLACE
STOP EDIT
END OF PROGRAM
CHANGE MODE
CARRIAGE RETURN
SPACE
PERIOD
COMMA
アイウエオカキクケコサシスセソタチツテトナニヌネノハヒフヘホマミムメモヤユヨラリルレロワヲンガギグゲゴザジズゼゾダヂヅデドバビブベボパピプペポー
ァィゥェォッャュョあいうえおかきくけこさしすせそたちつてとなにぬねのはひふへほまみむめもやゆよらりるれろわをんっゃゅょがぎぐげござじずぜぞだぢづ
でどばびぶべぼぱぴぷぺぽ

Fig. 5. The Japanese character table for the Rand tablet.

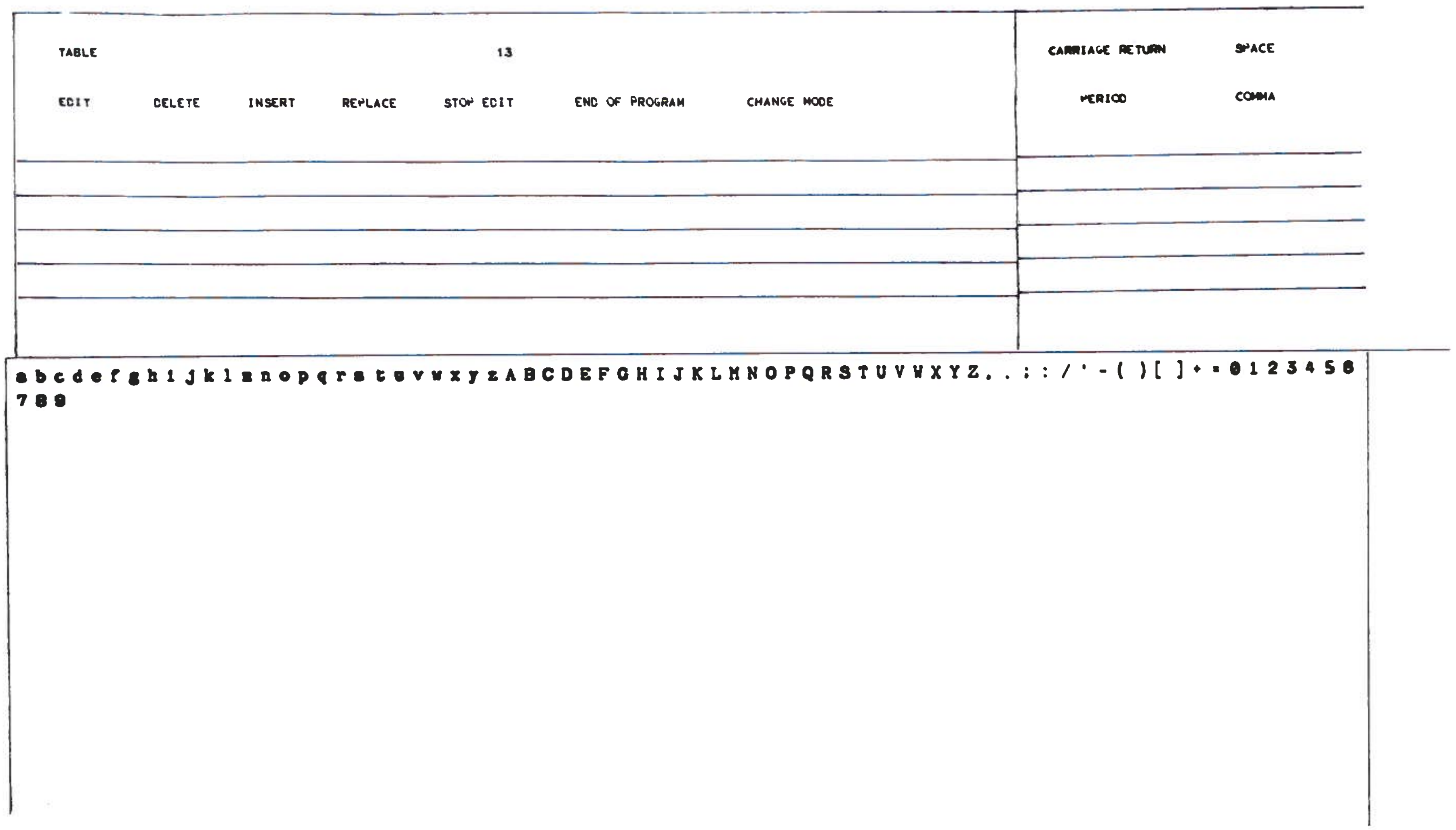


TABLE
13
EDIT
DELETE
INSERT
REPLACE
STOP EDIT
END OF PROGRAM
CHANGE MODE
CARRIAGE RETURN
SPACE
PERIOD
COMMA
a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z , . ; : / ' - ( ) [ ] + = 0 1 2 3 4 5 6
7 8 9

Fig. 6. The Rand table of alphabetic and special characters.

position $9 \times 72+11=659$ which contains the location pointer for this character. Then, by means of this pointer, the 16 pairs of co-ordinates of this character shown in the previous section are accessed for display and printing purposes.

The user may continue to encode characters from the current table until a text character is encountered which is not in the table. The table must then be replaced by the one which contains the character. The new table ID must be recorded, and encoding continues.

The text characters thus encoded are displayed on the scope. Should an error occur, the displayed text may be edited during the encoding process. When the user wishes to edit, he must select the appropriate editing control words seen in the Rand tablet character tables in Figures 2-6: EDIT, DELETE, INSERT, REPLACE, STOP EDIT.

3.4 *Character Entry*

If the text character about to be encoded does not appear in any of the Rand tables, then the user may enter the character entry mode. This is accomplished by selecting the control word CHANGE MODE seen in the Rand tables in the figures. A 16×16 grid appears on the scope and the user is able to draw the desired character on the scope grid with the Rand pen. This is accomplished by selecting grid points as the beginning and end points of straight lines which comprise the character. When the beginning and end points of a line have been selected, the line automatically appears on the scope. The user may assign indices (such as telecode, pronunciation, radical and stroke number) to the character if he so desires, via the scope and Rand pen.

The new character should then be manually written after the last printed character in the last Chinese table; when this new character is to be encoded in the text it must be accessed from this location. For example, new characters must be hand-written starting after the last printed Chinese character in Figure 3. The co-ordinate specifications and indices of new characters may either be outputted on magnetic tape and thus permanently stored in the system or may be used for the current run only.

Figure 3 is the last Chinese character table in our present system. Although we have the co-ordinate specifications of 8,000 on tape in our repertoire, due to PDP-1 storage size, only 4,100 Chinese characters may be conveniently stored simultaneously with Japanese, English and special characters. Were the system to become more widely in demand among researchers, it could, if necessary, quite easily be modified to access in one text any of the 8,000 Chinese characters, along with Japanese, English, Korean, Tamil, Pahlavic and Hebrew, and special characters. Pahlavic and Hebrew, Korean and Tamil are now contained in separate repertoires. If these diverse repertoires were combined in this fashion, it would certainly be preferable to provide the user with the facility for entering new characters into any location within any table of his choice. The current restriction upon entering the character into the first non-used slot in Figure 3 could easily be thus altered.

3.5 *Text Printing*

The encoded text is stored on magnetic tape and may be printed in the text-printing component on the S-C 4020. The S-C 4020 produces both a microfilm and a hard copy of the text at the speed of approximately 3 seconds per frame.[11] A hard copy can be used for xerox reproductions. A microfilm can be used as it is for photographic reproductions.

It is possible to vary the size of printed characters. Figure 14 is a printout of a Japanese mathematical text in size 1. Figure 13 is a printout of the same text in size 2, which is twice as large as size 1. Sizes 3, 4, etc. (3 times and 4 times as large as zise 1) are theoretically possible. In size 1, $56 \times 56=3136$ characters can be printed on one page or frame. In size 2, $28 \times 28=786$ characters can be printed. It is also possible to mix sizes in one text. For example, it is possible to produce a Chinese-English dictionary in which headings in Chinese are printed in size 2, and English translation in size 1, so that corresponding to each Chinese heading, two lines of English translation can be printed.

The S-C 4020 has the feature that the total density of each straight line is constant. The result of this is that a long line is printed thin, while a short line is printed thick. The printout of Chinese characters in size 1 tends to loom darker, with adjacent lines often merged into one another forming one bold line. This problem of resolution depends very much upon the condition of the built-in camera of the computer, which is not always as stable as one may wish. However, the original hard copy of a size 1 printout of Chinese characters is quite legible to native speakers of Chinese and Japanese, although xeroxed copies or photographically reproduced copies of the microfilm may have much poorer quality. Printouts of the characters in size 2 are always extremely legible.

[11] A frame is an image of text that can be displayed on the S-C 4020 scope at a time. The cost of microfilm and/or hard copy varies, but is somewhere between 50¢ and 75¢ per frame.

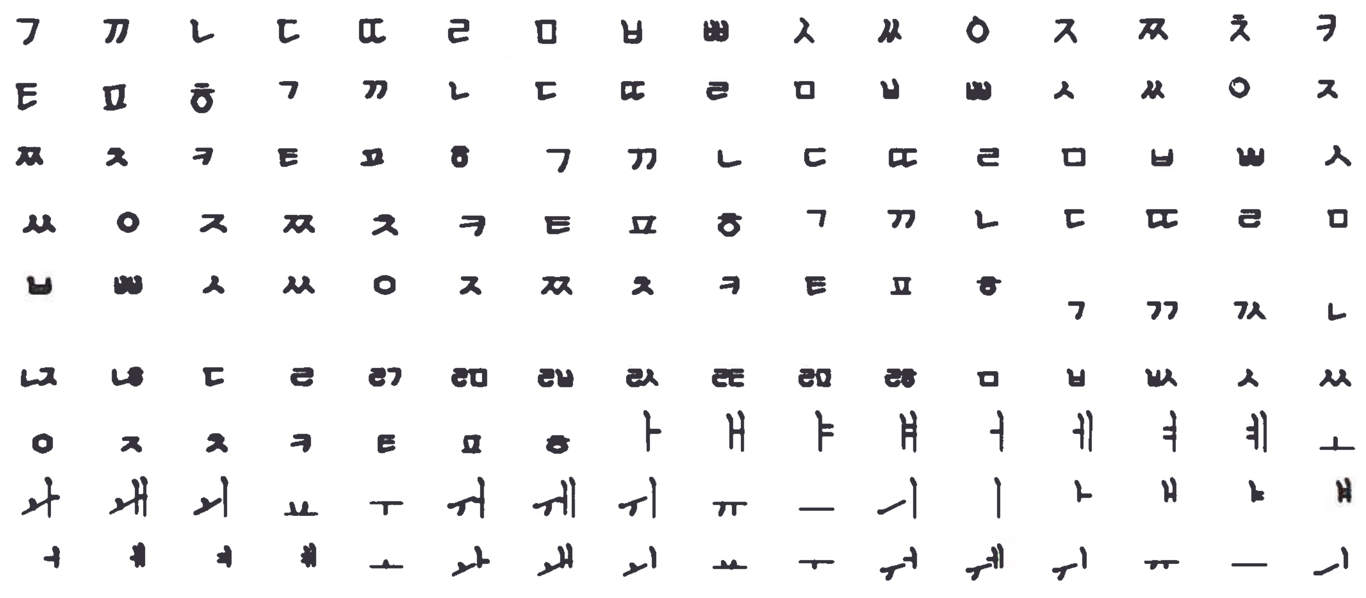


Fig. 7. S-C 4020 output of Korean component characters.

Sample S-C 4020 output from the current version of the system may be found in Figures 7-13. The various orthographies which our system can handle are illustrated.

The basic principle of character storage is the same for all orthographies; however, there have been some modifications for Korean and Tamil. In general, a character is stored as a unique self-contained unit consisting of the co-ordinate data for that character. In the Korean and Tamil alphabets, however, each alphabetic character represents a syllable containing up to three component characters. Thus Tamil and Korean both lend themselves to a more efficient means of character storage. In both cases, only the component characters have the associated co-ordinate specifications, while the alphabetic characters (syllables) are represented as combinations of particular components. For example, in the case of Korean, only the 145 component characters seen in Figure 7 have co-ordinate data stored for them. The 2,000 syllabic characters which comprise the Korean orthography, a portion of which appears in Figure 8, are internally represented as combinations of particular components. Thus the actual co-ordinate information for a given component configuration is stored only once, and is not repeated each time the component character occurs in a syllabic character.

Whereas in the Korean orthography, the two or three component characters which comprise a syllabic character are superimposed one upon another, in Tamil a syllabic character may consist of one, two or three components which are written side by side, from left to right. A portion of the Tamil alphabet may be seen in Figure 9. In the Tamil orthography, although there is no special spacing between the syllabic characters that form a word, no syllabic character may be divided at the end of a line. The S-C 4020 output of a Tamil text in Figure 10 incorporates this feature.

The Hebrew and Pahlavic texts in Figures 11 and 12 are printed from right to left. The Japanese text in Figure 13 which contains Chinese, Japanese, English, and special mathematical characters is printed from left to right. A vertical printout of the same text may also be obtained by simply changing a command instruction for printing.

3.6 *Use of the System for Text Printing Only*

The text encoding and the character entry operations described in §§3.2 and 3.3 assume the availability of a PDP-1 computer with attached display scopes and a Rand tablet. Therefore, these components can be used only at Harvard or other places which have the same computer configuration. Moreover, even when the computer is available, it is usually the case that the user cannot monopolize the facilities for an extended period of time. Therefore, when a large amount of text is to be encoded, it is difficult to input it via the Rand tablet on-line. In such a case, our system can be used simply as a printing device, with the encoding of the text conducted manually off-line. Each character in the text is replaced by its unique alphanumeric code (four digit number in the case of Chinese characters), which is

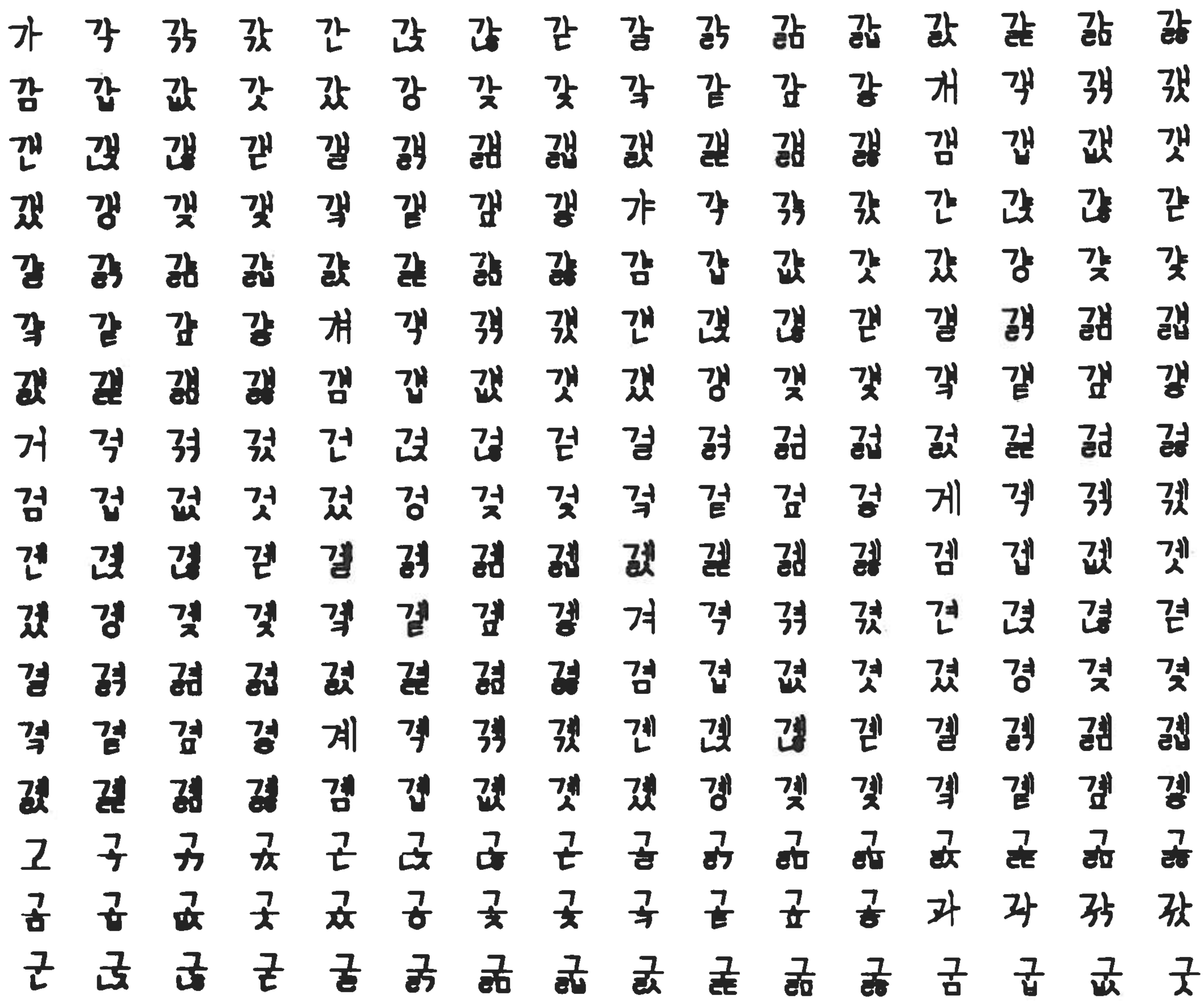
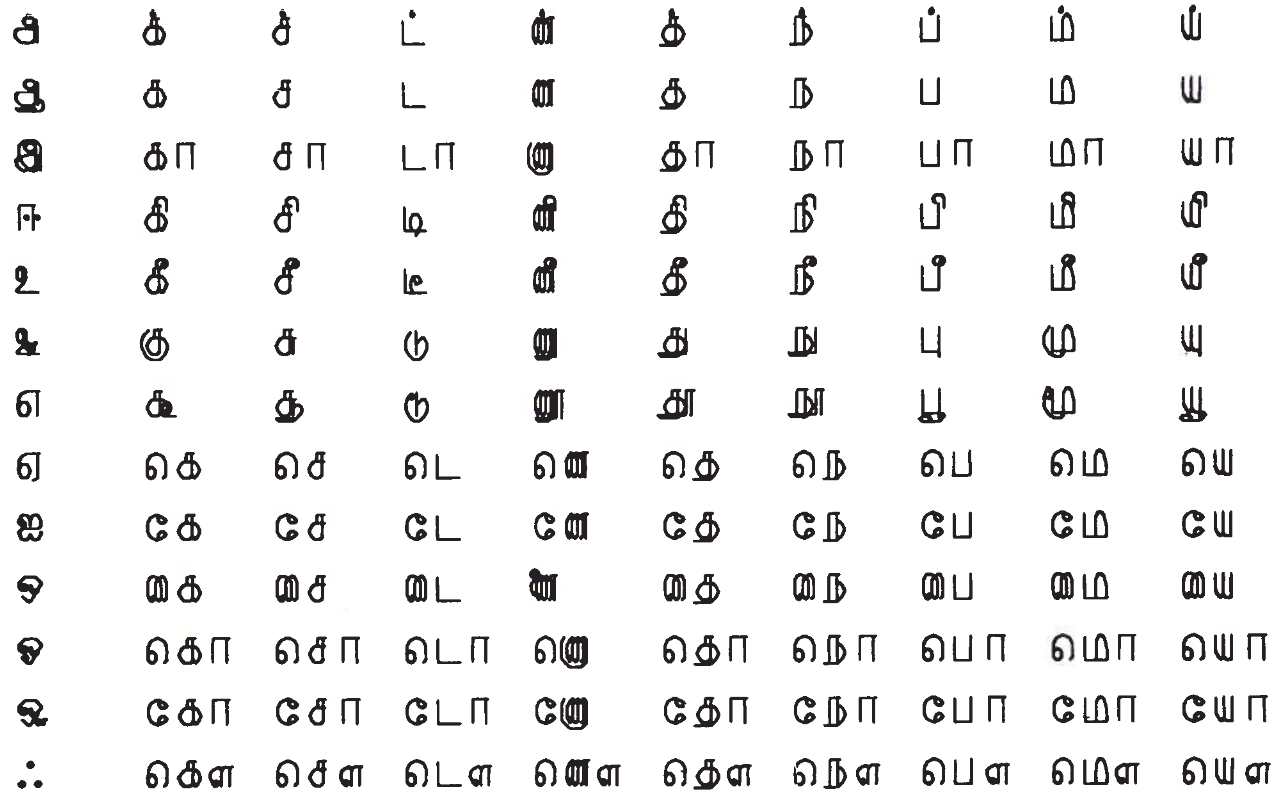


Fig. 8. S-C 4020 output of a sample of Korean syllabic characters.

recorded on punch cards. Magnetic tape thus produced, consisting of a sequence of alphanumeric codes is then compared with the dictionary of co-ordinate specifications, with each code translated into pairs of coordinates representing the character. This program is operational on an IBM 7094. The tape of coordinate data is then run on an S-C 4020 for printing.

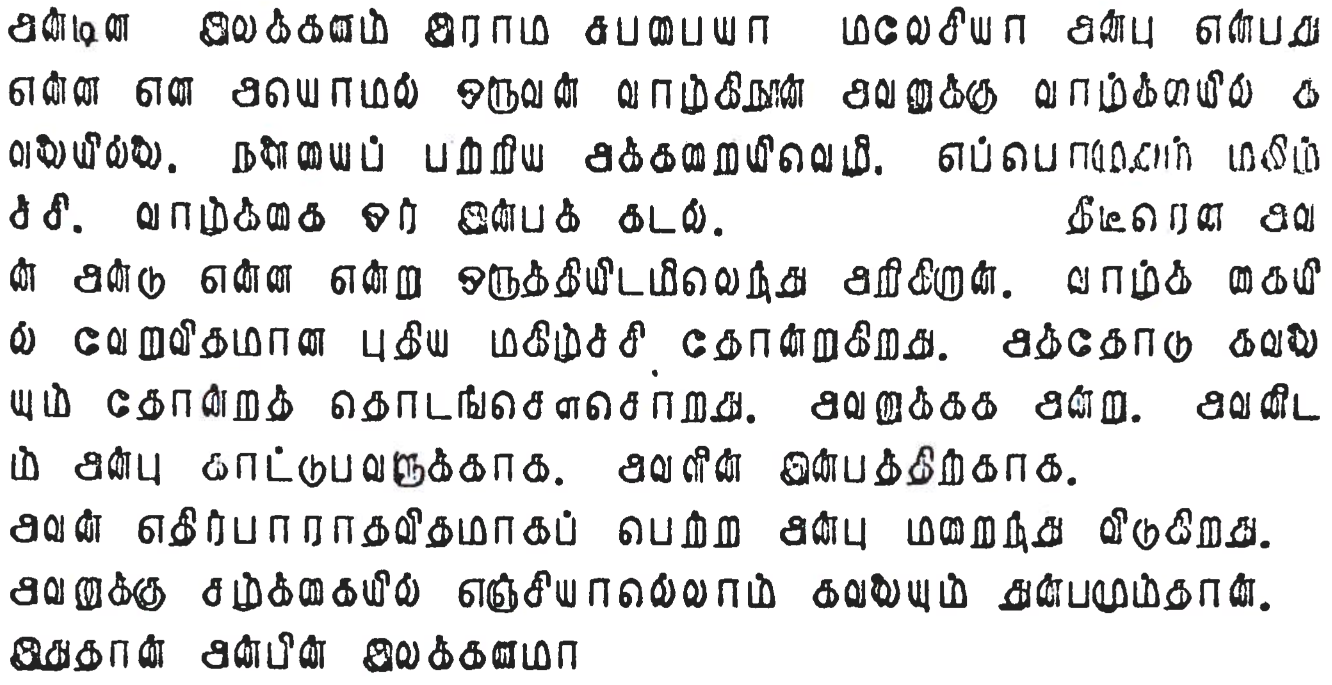
For orthographies such as Korean and Tamil, transliteration systems more or less based on pronunciations are now being designed which will preserve a one to one correspondence between transliteration and original characters. When these systems are completed, the original text will be punched on cards in the legible and pronounceable transliteration, which will be translated into co-ordinate specifications of each original character for an S-C 4020 printout. Programs for printing the original text both in transliteration and the original orthography, inter-lineally, on alternate pages, and independently, are also being planned.

Since the conversion program from alphanumeric codes to co-ordinate specifications is written in Fortran IV, and since the conversion programs from transliteration to co-ordinate specifications will also be written in Fortran IV, and can be run on almost all standard computers, and since an S-C 4020 is a commercial computer accessible at several commercial and research institutions at cost the text printing capability of our system is and will be of the most general utility, involving no hardware investment and



அ	க்	ச்	ட்	ண்	த்	ந்	ப்	ம்	ய்
ஆ	க	ச	ட	ண	த	ந	ப	ம	ய
இ	கா	சா	டா	ணா	தா	நா	பா	மா	யா
ஈ	கி	சி	டி	ணி	தி	நி	பி	மி	யி
உ	கீ	சீ	டீ	ணீ	தீ	நீ	பீ	மீ	யீ
ஊ	கு	சு	டு	ணு	து	நு	பு	மு	யு
எ	கூ	சூ	டூ	ணூ	தூ	நூ	பூ	மூ	யூ
ஏ	கெ	செ	டெ	ணெ	தெ	நெ	பெ	மெ	யெ
ஐ	கே	சே	டே	ணே	தே	நே	பே	மே	யே
ஒ	கை	சை	டை	ணை	தை	நை	பை	மை	யை
ஓ	கொ	சொ	டொ	ணொ	தொ	நொ	பொ	மொ	யொ
ஔ	கோ	சோ	டோ	ணோ	தோ	நோ	போ	மோ	யோ
ஃ	கௌ	சௌ	டௌ	ணௌ	தௌ	நௌ	பௌ	மௌ	யௌ

Fig. 9. S-C 4020 output of part of the Tamil syllabic alphabet.



அன்டின இலக்கணம் இராம சபபையா மலேசியா அன்பு என்பது
என்ன என அயொமல் ஒருவன் வாழ்கிறான் அவனுக்கு வாழ்க்கையில் க
வலையில்லை. நளையைப் பற்றிய அக்கறையிவெழி. எப்பொ[illegible] மகிழ்
ச்சி. வாழ்க்கை ஓர் இன்பக் கடல். திடீரென அவ
ன் அன்டு என்ன என்று ஒருத்தியிடமிலெந்து அறிகிறான். வாழ்க் கையி
ல் வேறுவிதமான புதிய மகிழ்ச்சி தோன்றுகிறது. அத்தோடு கவலை
யும் தோன்றத் தொடங்கெளசொறது. அவனுக்கக அன்று. அவனிட
ம் அன்பு காட்டுபவருக்காக. அவளின் இன்பத்திற்காக.
அவன் எதிர்பாராதவிதமாகப் பெற்ற அன்பு மறைந்து விடுகிறது.
அவனுக்கு சழ்க்கையில் எஞ்சியாவெல்லாம் கவலையும் துன்பமும்தான்.
இதுதான் அன்பின் இலக்கணமா

Fig. 10. S-C 4020 output of a sample Tamil text.

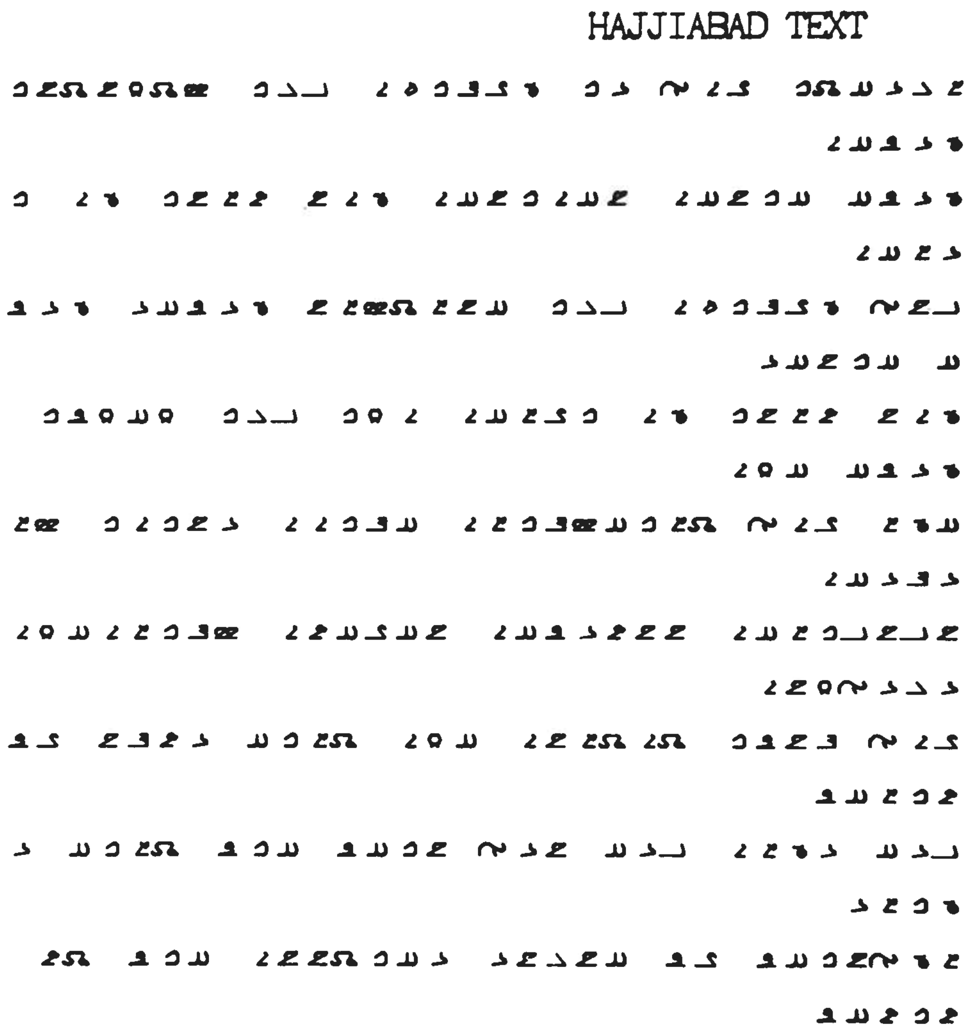

HAJJIABAD TEXT

Fig. 11. S-C 4020 output of a sample Pahlavic text.

אורלי ילדה חמודה מאד כבר נ

שדו לה שתי שיניים

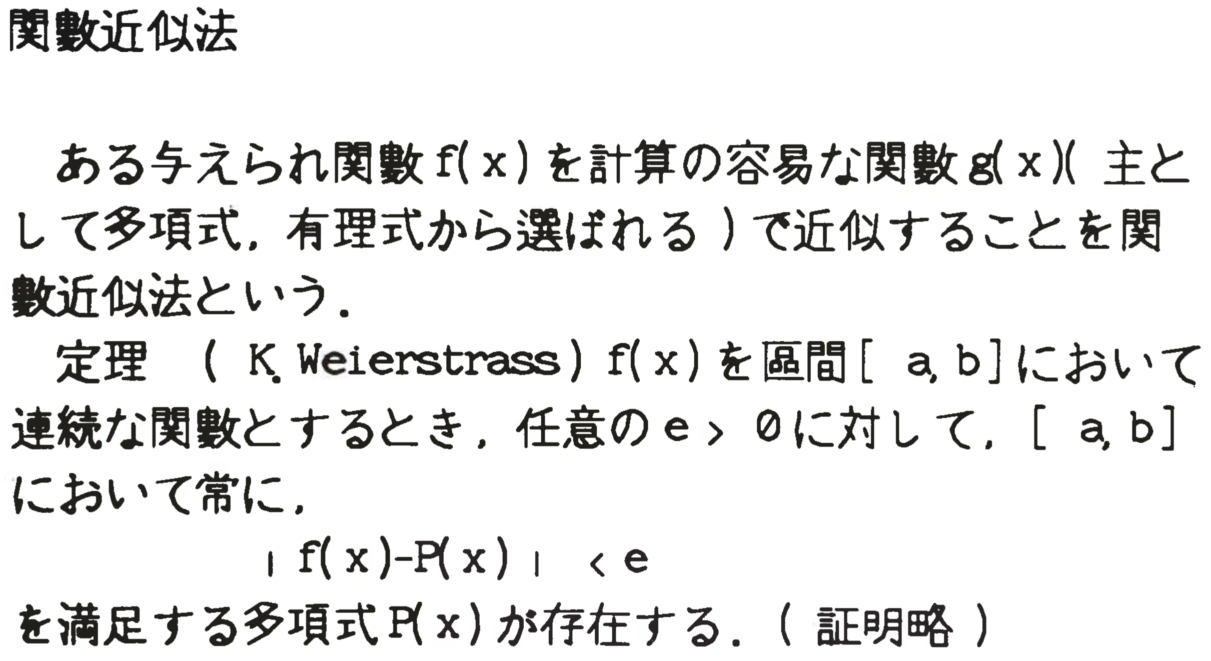
Fig. 12. S-C 4020 output of a Hebrew text.

little software development for possible users of our system.

3.7 *Possible Applications*

Our system, especially its text printing component, provides researchers using non-standard orthographies with computer printouts of their data and analysis, such as concordances of texts, in their original orthographies. Computer compilation and editing of dictionaries involving non-standard orthographies are also made possible, and if one is willing to sacrifice a certain degree of elegance of calligraphy, S-C 4020-generated microfilms can be used as they are for photographic reproductions of as many copies as one may need for publication and distribution.

One possible avenue of application of our system is the computer production of library cards for foreign books and journals. A pilot study is now being conducted at Harvard for producing microfilms with four library card images per frame as seen in Figure 15. A microfilm print-


関數近似法

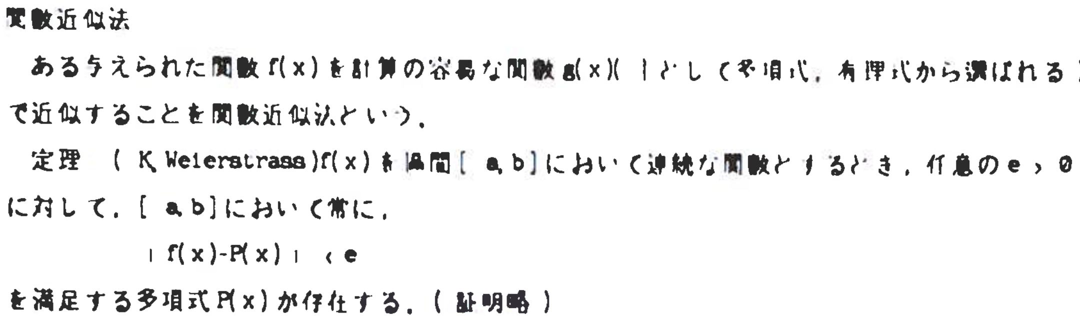
ある与えられ関數 f(x) を計算の容易な関數 g(x)(主として多項式，有理式から選ばれる) で近似することを関數近似法という．

定理　(K. Weierstrass) f(x) を區間 [a, b] において連続な関數とするとき，任意の e > 0 に対して，[a, b] において常に，

| f(x)-P(x) | < e

を満足する多項式 P(x) が存在する．(証明略)

Fig. 13. S-C 4020 output of a sample Japanese text.


関數近似法

ある与えられた関數 f(x) を計算の容易な関數 g(x)(主として多項式，有理式から選ばれる)
で近似することを關數近似法という．

定理　(K. Weierstrass)f(x) を區間 [a, b] において連続な関數とするとき，任意の e > 0
に対して，[a, b] において常に，

| f(x)-P(x) | < e

を満足する多項式 P(x) が存在する．(証明略)

Fig. 14. Printout of the same text in size 1.

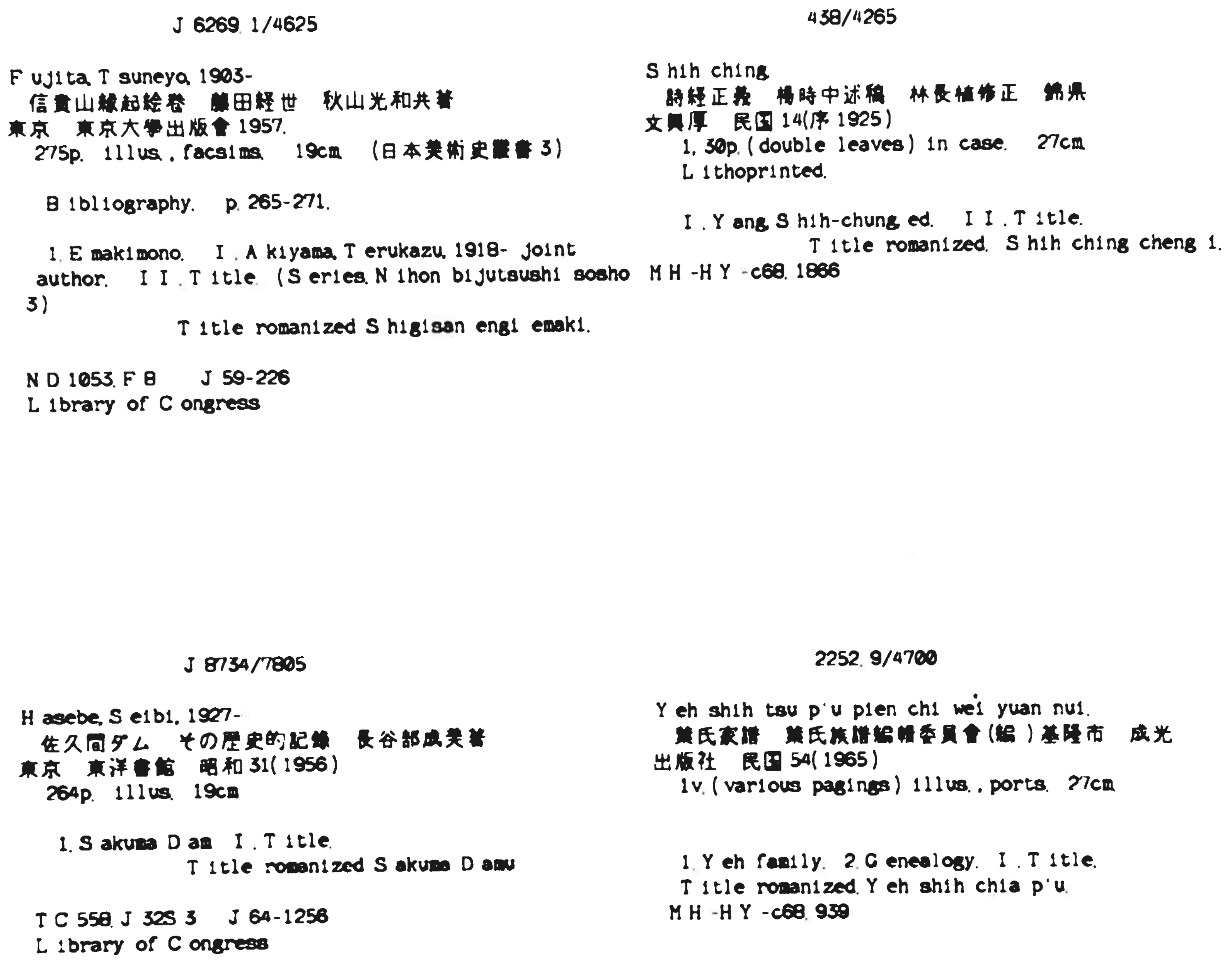
er will produce as many copies of library cards as needed, and library catalogues of books and journals on authors, titles, and subject indexes can be readily produced as punched cards for acquisitions accumulated.

It is also planned to add to the repertoire of our system as many non-standard orthographies as there are users for them.

4. ACKNOWLEDGEMENTS

The authors wish to extend their gratitude to Richard See, National Library of Medicine, Gordon L. Walker and Barbara N. Smith of the American Mathematical Society, Roland B. Holt of Device Development Corporation, Frank A. Kierman, Jr. of Princeton University, Edward Wagner of Harvard University, Rama Subbiah of the University of Malaya, John Moyne of IBM, Daniel Cohen of Harvard University, Hideyuki Hayashi of the Data Center, Ministry of International Trade and Industry, Japan, for their invaluable advice and assistance in the implementation of the system at various stages of its development.

The project was started at the suggestion of David Packard, Jr., University of California at Los Angeles, who experimented on the S-C 4020 printing of Greek, Minoan, and Hieroglyphic scripts while he was at Harvard. The use of Stromberg-Carlson's 4020/4060 was conceived of



J 6269.1/4625

F ujita, T suneyo, 1903-
信貴山緣起絵卷 藤田經世 秋山光和共著
東京 東京大學出版會 1957.
275p. illus., facsims. 19cm (日本美術史叢書 3)

B ibliography. p. 265-271.

1. E makimono. I. A kiyama, T erukazu, 1918- joint author. I I. T itle. (S eries. N ihon bijutsushi sosho 3)
T itle romanized: S higisan engi emaki.

N D 1053. F 8 J 59-226
L ibrary of C ongress

438/4265

S hih ching.
詩經正義 楊時中述稿 林長植修正 錦県
文興厚 民国 14(序 1925)
1, 30p. (double leaves) in case. 27cm
L ithoprinted.

I. Y ang, S hih-chung, ed. I I. T itle.
T itle romanized: S hih ching cheng i.
M H -H Y -c68. 1866

J 8734/7805

H asebe, S eibi, 1927-
佐久間ダム その歴史的記錄 長谷部成美著
東京 東洋書館 昭和 31(1956)
264p. illus. 19cm

1. S akuma D am. I. T itle.
T itle romanized: S akuma D amu

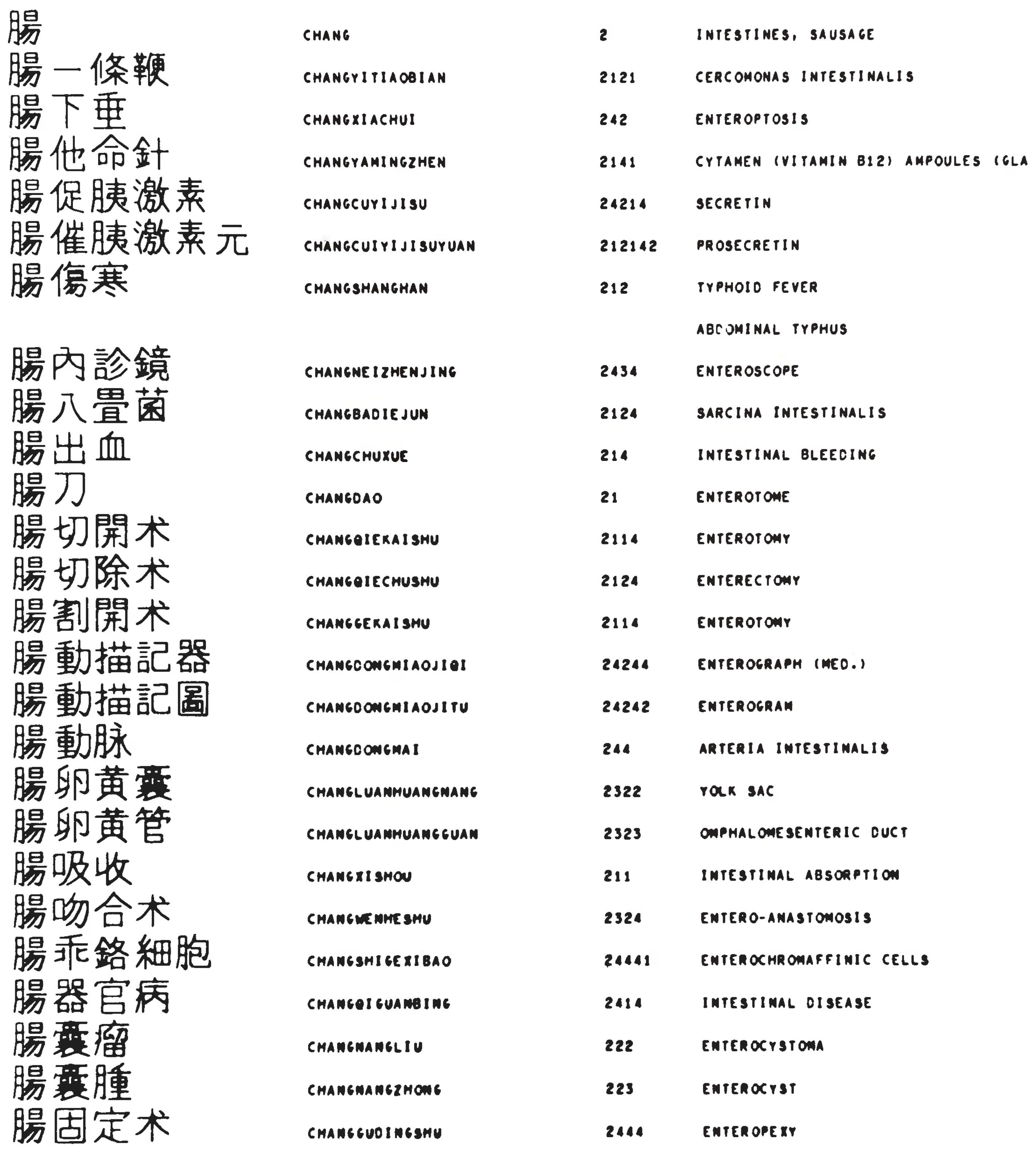
T C 558. J 32S 3 J 64-1258
L ibrary of C ongress

2252.9/4700

Y eh shih tsu p'u pien chi wei yuan hui.
葉氏家譜 葉氏族譜編輯委員會(編)基隆市 成光出版社 民国 54(1965)
1v. (various pagings) illus., ports. 27cm

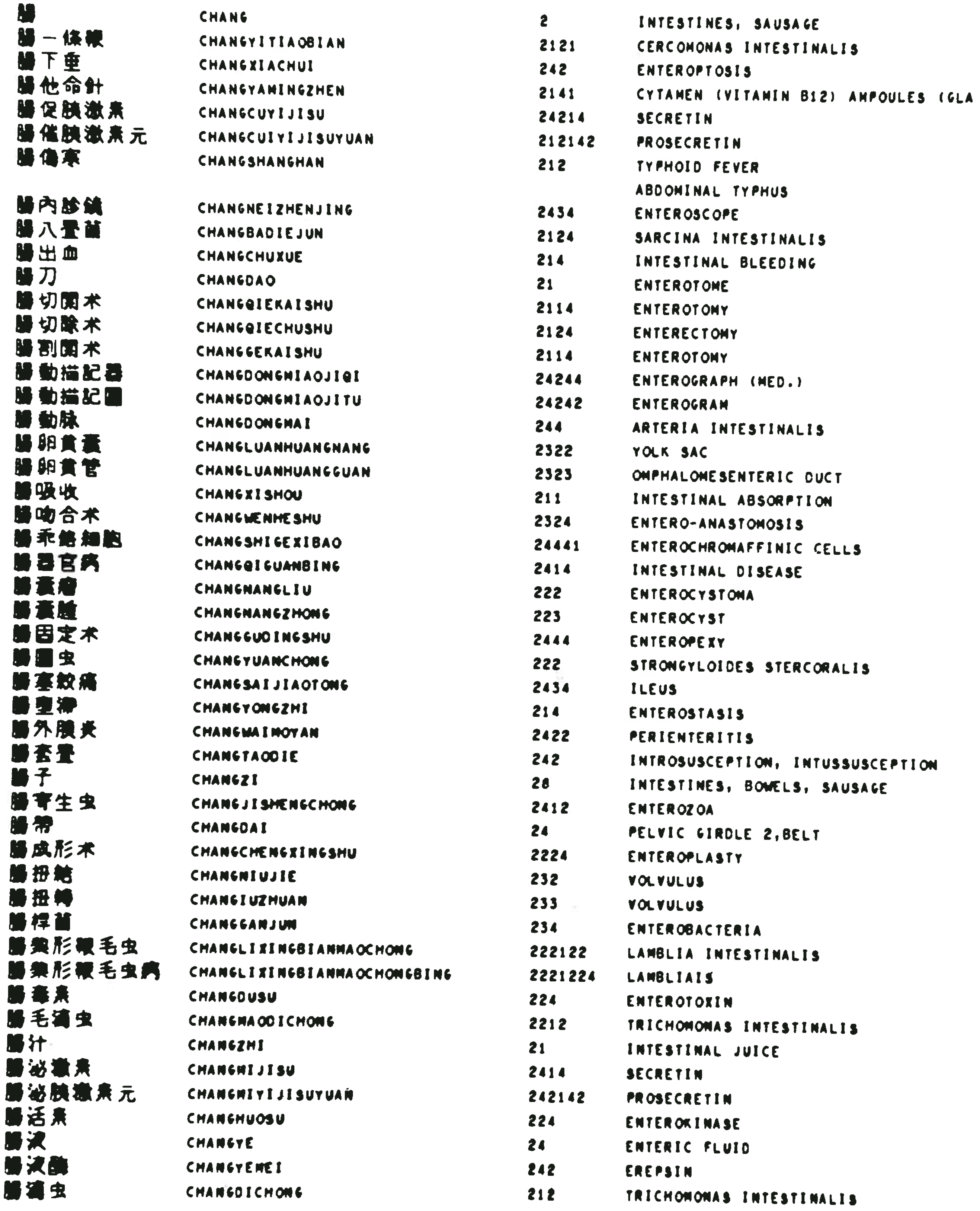
1. Y eh family. 2. G enealogy. I. T itle.
T itle romanized: Y eh shih chia p'u.
M H -H Y -c68. 939

Fig. 15. An S-C 4020 frame with four library card images.



腸	CHANG	2	INTESTINES, SAUSAGE
腸一條鞭	CHANGYITIAOBIAN	2121	CERCOMONAS INTESTINALIS
腸下垂	CHANGXIACHUI	242	ENTEROPTOSIS
腸他命針	CHANGYAMINGZHEN	2141	CYTAMEN (VITAMIN B12) AMPOULES (GLA
腸促胰激素	CHANGCUYIJISU	24214	SECRETIN
腸催胰激素元	CHANGCUIYIJISUYUAN	212142	PROSECRETIN
腸傷寒	CHANGSHANGHAN	212	TYPHOID FEVER
			ABDOMINAL TYPHUS
腸內診鏡	CHANGNEIZHENJING	2434	ENTEROSCOPE
腸八疊菌	CHANGBADIEJUN	2124	SARCINA INTESTINALIS
腸出血	CHANGCHUXUE	214	INTESTINAL BLEEDING
腸刀	CHANGDAO	21	ENTEROTOME
腸切開术	CHANGQIEKAISHU	2114	ENTEROTOMY
腸切除术	CHANGQIECHUSHU	2124	ENTERECTOMY
腸割開术	CHANGGEKAISHU	2114	ENTEROTOMY
腸動描記器	CHANGDONGMIAOJIQI	24244	ENTEROGRAPH (MED.)
腸動描記圖	CHANGDONGMIAOJITU	24242	ENTEROGRAM
腸動脉	CHANGDONGMAI	244	ARTERIA INTESTINALIS
腸卵黃囊	CHANGLUANHUANGNANG	2322	YOLK SAC
腸卵黃管	CHANGLUANHUANGGUAN	2323	OMPHALOMESENTERIC DUCT
腸吸收	CHANGXISHOU	211	INTESTINAL ABSORPTION
腸吻合术	CHANGWENHESHU	2324	ENTERO-ANASTOMOSIS
腸乖鉻細胞	CHANGSHIGEXIBAO	24441	ENTEROCHROMAFFINIC CELLS
腸器官病	CHANGQIGUANBING	2414	INTESTINAL DISEASE
腸囊瘤	CHANGNANGLIU	222	ENTEROCYSTOMA
腸囊腫	CHANGNANGZHONG	223	ENTEROCYST
腸固定术	CHANGGUDINGSHU	2444	ENTEROPEXY

Fig. 16-1. S-C 4020 output of a sample text from the *Chinese-English Dictionary* (McGraw-Hill).



腸	CHANG	2	INTESTINES, SAUSAGE
腸一條鞭	CHANGYITIAOBIAN	2121	CERCOMONAS INTESTINALIS
腸下垂	CHANGXIACHUI	242	ENTEROPTOSIS
腸他命針	CHANGYAMINGZHEN	2141	CYTAMEN (VITAMIN B12) AMPOULES (GLA
腸促胰激素	CHANGCUYIJISU	24214	SECRETIN
腸催胰激素元	CHANGCUIYIJISUYUAN	212142	PROSECRETIN
腸傷寒	CHANGSHANGHAN	212	TYPHOID FEVER
			ABDOMINAL TYPHUS
腸內診鏡	CHANGNEIZHENJING	2434	ENTEROSCOPE
腸八疊菌	CHANGBADIEJUN	2124	SARCINA INTESTINALIS
腸出血	CHANGCHUXUE	214	INTESTINAL BLEEDING
腸刀	CHANGDAO	21	ENTEROTOME
腸切開术	CHANGQIEKAISHU	2114	ENTEROTOMY
腸切除术	CHANGQIECHUSHU	2124	ENTERECTOMY
腸割開术	CHANGGEKAISHU	2114	ENTEROTOMY
腸動描記器	CHANGDONGMIAOJIQI	24244	ENTEROGRAPH (MED.)
腸動描記圖	CHANGDONGMIAOJITU	24242	ENTEROGRAM
腸動脈	CHANGDONGMAI	244	ARTERIA INTESTINALIS
腸卵黃囊	CHANGLUANHUANGNANG	2322	YOLK SAC
腸卵黃管	CHANGLUANHUANGGUAN	2323	OMPHALOMESENTERIC DUCT
腸吸收	CHANGXISHOU	211	INTESTINAL ABSORPTION
腸吻合术	CHANGWENHESHU	2324	ENTERO-ANASTOMOSIS
腸嗜鉻細胞	CHANGSHIGEXIBAO	24441	ENTEROCHROMAFFINIC CELLS
腸器官病	CHANGQIGUANBING	2414	INTESTINAL DISEASE
腸囊瘤	CHANGNANGLIU	222	ENTEROCYSTOMA
腸囊腫	CHANGNANGZHONG	223	ENTEROCYST
腸固定术	CHANGGUDINGSHU	2444	ENTEROPEXY
腸圓虫	CHANGYUANCHONG	222	STRONGYLOIDES STERCORALIS
腸塞絞痛	CHANGSAIJIAOTONG	2434	ILEUS
腸壅滯	CHANGYONGZHI	214	ENTEROSTASIS
腸外膜炎	CHANGWAIMOYAN	2422	PERIENTERITIS
腸套疊	CHANGTAODIE	242	INTROSUSCEPTION, INTUSSUSCEPTION
腸子	CHANGZI	28	INTESTINES, BOWELS, SAUSAGE
腸寄生虫	CHANGJISHENGCHONG	2412	ENTEROZOA
腸帶	CHANGDAI	24	PELVIC GIRDLE 2,BELT
腸成形术	CHANGCHENGXINGSHU	2224	ENTEROPLASTY
腸扭結	CHANGNIUJIE	232	VOLVULUS
腸扭轉	CHANGIUZHUAN	233	VOLVULUS
腸桿菌	CHANGGANJUN	234	ENTEROBACTERIA
腸梨形鞭毛虫	CHANGLIXINGBIANMAOCHONG	222122	LAMBLIA INTESTINALIS
腸梨形鞭毛虫病	CHANGLIXINGBIANMAOCHONGBING	2221224	LAMBLIAIS
腸毒素	CHANGDUSU	224	ENTEROTOXIN
腸毛滴虫	CHANGMAODICHONG	2212	TRICHOMONAS INTESTINALIS
腸汁	CHANGZHI	21	INTESTINAL JUICE
腸泌激素	CHANGMIJISU	2414	SECRETIN
腸泌胰激素元	CHANGMIYIJISUYUAN	242142	PROSECRETIN
腸活素	CHANGHUOSU	224	ENTEROKINASE
腸液	CHANGYE	24	ENTERIC FLUID
腸液酶	CHANGYEMEI	242	EREPSIN
腸滴虫	CHANGDICHONG	212	TRICHOMONAS INTESTINALIS

Fig. 16-2. S-C 4020 output of a sample text from the *Chinese-English Dictionary* (McGraw-Hill).

independently by two other groups around the same time (the summer of 1967). A. V. Hershey's system (Hershey, 1967) is very similar to our text-printing component, except that he uses a larger grid, and that he takes advantage of S-C 4060's feature of specifying the width of each straight line in four degrees. His system produces Chinese and Japanese character printouts whose style approaches that of artistic brush-stroke calligraphy. The repertoire of characters of his sytem is unfortunately very limited.

June 1969

REFERENCES

[1l Anonymous, *Introduction and Operation of the Fully Automatic Chinese Monotype of the United Daily News Model* (Taipei, Taiwan, September 1965).

[2] C. Y. Dougherty, S. M. Lamb, and S. E. Martin, *Chinese Character Indexes* (Berkeley, California, University of California Press, 1963).

[3] H. Hayashi, S. Duncan, and S. Kuno, "Graphical Input/Output of Non-Standard Characters", *Communications of the ACM*, Vol. 11, Number 9 (September 1968).

[4] H. Hayashibara and K. Saito, "Kanji Display System", *Japan Electronic Engeneering*, September 1968.

[5] A. V. Hershey, "Calligraphy for Computers", U.S. Naval Weapons Laboratory, *Technical Report*, No. 2101 (Dahlgren, Virginia, 1967).

[6] F. A. Kierman, Jr., and E. Barber, "Computers and Chinese Linguistics", *Unicorn*, No. 3 (Princeton University, Chinese Linguistics Project and Seminar, 1968), pp. 29-73.

[7] F. E. Shashoua, "Photocomposition Machine for the Chinese Language", *RCA, Camden, New Jersey, Technical Paper* (1964).

[8] G. L. Walker, S. Kuno, B. N. Smith, and R. B. Holt, "Chinese Mathematical Text Analysis", *IEEE Transactions on Engineering, Writing and Speech*, Vol. EWS-11, Number 2 (August 1968).

A Computer Graphics System for Variable Series Orthographies

CARL LEBAN and CHARLES T. BAIRD
The University of Kansas at Lawrence

The first step toward implementing automatic processing in the study of the world's languages must be the development of automatic methods of manipulating their graphic systems. An unconventional definition of alphabet *is presented here, one which denies the phonemic nature of alphabets and the concept of the* morphemic *writing system. What is asserted is that at the graphemic level all writing systems are characterized by* small finite sets *of graphic signs (alphabets). Elsewhere we have described a theoretical rationale for automated synthesis of the complex polyalternating polyvariable series, Chinese. This paper describes an exercise in process of our work with Chinese in which we have achieved a perfect and direct synthesis of a bialternating polyvariable script, Korean* hangul.

If the advantages of automatic processing are to be achieved for any and all of the world's languages, then a necessary first step must be the achievement of automatic methods of manipulating the graphic systems of those languages. As obvious as the foregoing truism is the fact that ideally, such systems ought to be direct, preserving all inherent graphic features throughout the automation process, and neither distorting nor obscuring them.[1] Such systems will necessarily be synthetic, since all 'natural' orthographic systems develop their graphs by synthesis from constituent elements, and this process of synthesis itself is an inherent and sometimes crucial graphic property.

The literature to date has been characterized by an unfortunate fuzziness of terminology regarding writing systems, to the extent that non-Roman orthographies are sometimes lumped wholesale under such descriptions as 'non-alphabetic', and even 'non-standard'. Aside from the chauvinism implied in such terminology, their imprecision prevents a clear apprehension of the true nature of such scripts, and thus the true nature of the problems met in attempting their automation.

We define *alphabet* as follows: A small finite set of graphic signs which may or may not have phonetic, semantic, or other value, and which are used singly or in combination according to predetermined orthographic rules, to write the graphs of one or more languages.[2]

This definition will be startling to the conventional linguist, on the one hand denying that alphabets are exclusively phonemic representations, and on the other hand denying the whole concept of so-called *morphemic* writing systems. What we dare to assert here is that *all* writing systems are characterized at their graphemic level by *small finite sets* of graphic signs, which sets we prefer to call alphabets. Moreover, since every writing system is an effective means of synthesizing graphs from the signs of the alphabet, orthographic rules are essentially sets of criteria for determining proper serial order of those graphic signs, upon which further subsets of phonetic, semantic, graphic and other conventions are superimposed. Though the rules for determining serial order may vary from the simplest possible to extremely complex, none of them, of course, are random, and precise definition of the rule sets provides a basis for a rigorous though necessarily tentative typology, as follows:

I. INVARIABLE. This is the simplest possible method of combination, in which successive graphic signs of the alphabet are arranged in a simple invariable series, as from left to right, top to bottom, and so forth. Examples: Roman, Greek, Cyrillic, Japanese *kana*.

II. VARIABLE. These may be classified into subtypes:
A. MONO-ALTERNATING BIVARIABLE. For example, Hebrew, in which an invariable right to left series of

[1] Actually, there are five essential properties for an *ideal* automation of any graphic system: (1) Machine compatibility for read-in, (2) perfect graphic discretion, (3) no pre-storing requirement, (4) no limitation on corpus, and (5) printout in normal orthography.

[2] *Webster's Third New International Dictionary* (1962) defines orthography as "a method of representing the sounds of a language by written or printed symbols ..." and alphabet as "any set of characters with which one or more languages are written whether such characters are letters ..., the signs of a syllabary, or other basic units of writing ..." (definition 1b).

one subset of graphic signs (generally consonants) alternates with top to bottom juxtaposition of a second subset of graphic signs (generally vowels).

B. Mono-alternating polyvariable. For example, Sanskrit, in which an invariable left to right series of one subset of graphic signs (consonants) alternates with several different subset determined juxtapositions (left to right, right to left, top to bottom, bottom to top) for each of several other subsets of graphic signs (vowels).

C. Bialternating polyvariable. For example, Korean *hangul*, in which one subset of graphic signs (consonants) alternates between two juxtapositional modes (left to right, top to bottom) as determined by the presence in the graphic environment of a member or members of one or the other or both of two other subsets of graphic signs ('vertical' vowels, 'horizontal' vowels).

D. Polyalternating polyvariable. For example, Chinese, in which a number of undefined subsets of graphic signs combine with each other in any of several possible juxtapositional modes according to rules not yet fully elucidated.

The classification scheme proposed above does not pretend to be all-encompassing; indeed our knowledge of writing systems is too limited to so presume. Nonetheless, such a descriptive typology does provide us with a rationale for approaching the automation problem efficiently, by pointing up the difference in mode of synthesis in the various systems so that an attack on simulation may be rationally begun.

Of course, automation of invariable series orthographies already has a long history of success. Elsewhere,[3] we have described a theoretical rationale, presently in a well-developed experimental state, for automated synthesis of the very complex polyalternating polyvariable series, Chinese. As an exercise in process of our work with Chinese we have achieved a perfect and direct synthesis of a bialternating polyvariable script, Korean *hangul*.

The *hangul* script of Korea is perhaps the only writing system in the history of the world which neither evolved, nor was derived nor borrowed from any other. During the fifteenth century, a committee of scholars, fully cognizant of the difficulties of continuing to employ Chinese as a writing system for the completely unrelated Korean language, set to work to devise a phonetic script which would be uniquely suited to representation of the Korean sound system of that time. Their creation, the *hangul* alphabet, was so elegantly simple, so phonologically sound, and so esthetically beautiful that with a minimum of orthographic convention it continues to serve its intended function to the present day.

Historically, the influence of Korea's close neighbor and political suzereign, China, was so great that Chinese writing continued in a predominant position for five centuries. It was not until the mid-twentieth century, after the end of the second World War and the lifting of the Japanese colonial yoke that resurgent nationalism moved *hangul* into a position of predominance. Today, Chinese characters have been almost completely expunged from the writing system used in North Korea, and the trend in the south is to employ as few Chinese characters as possible in a mixed orthography similar to the Japanese method.

The *hangul* alphabet in current use consists of 28 phonetic signs, of which 14 are consonants and 14 vowels. The vowels in turn are seen to be divided into two further subsets: nine 'vertical' vowels and five 'horizontal' vowels. Fig. 1 displays these basic sets.

Hangul consonant	Yale romanization	*Hangul* vowel	Yale romanization
ㄱ	K	ㅏ	A
ㄴ	N	ㅐ	AY
ㄷ	T	ㅑ	YA
ㄹ	L	ㅒ	YAY
ㅁ	M	ㅓ	E
ㅂ	P	ㅔ	EY
ㅅ	S	ㅕ	YE
ㅇ	NG	ㅖ	YEY
ㅈ	C	ㅗ	O
ㅊ	CH	ㅛ	YO
ㅋ	KH	ㅜ	WU
ㅌ	TH	ㅠ	YU
ㅍ	PH	ㅡ	U
ㅎ	H	ㅣ	I

Fig. 1. The *hangul* alphabet.

In addition to the basic sets just described, a subset of five consonants is conventionally doubled in writing to represent the so-called "third series"[4] of stops in spoken Korean. The doubled consonants are written by doubling the single graphic sign in single sign space, as shown in Fig. 2. The only other purely graphic convention consists

[3] Carl Leban, "Synthetic Chinese", *Toward an Automated Comprehensive East Asian Bibliographic System, Special Conference Supplement to Behavioral Science Notes*, Vol. 4, No. 1 (1969), pp. 63-67.

[4] "Korean has been analyzed as having three series of stops. One is simple, with both voiced and voiceless allophones. One is aspirate and voiceless. Linguists have not been able to reach any agreement of the description of the third. It has been described as very fortis, as glottalized, or as merely doubled." H. A. Gleason Jr., *An Introduction to Descriptive Linguistics* (New York, 1961), p. 332.

of a single allograph, ㅜ for ㅜ in the diphthong environments: ㅝ, ㅞ and ㅟ.

Hangul single consonant	Yale romanization	*Hangul* double consonant	Yale romanization
ㄱ	K	ㄲ	KK
ㄷ	T	ㄸ	TT
ㅂ	P	ㅃ	PP
ㅅ	S	ㅆ	SS
ㅈ	C	ㅉ	CC

Fig. 2. Doubling of *hangul* consonants.

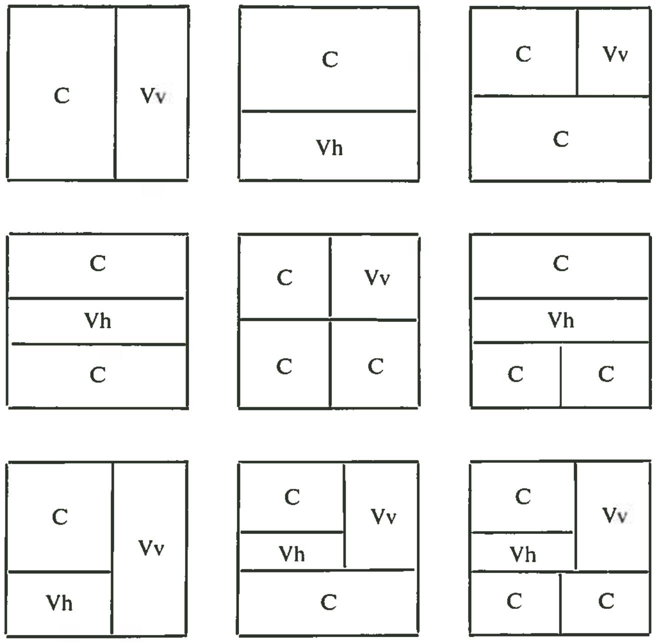
The rules for combining signs in writing are extremely simple and unexceptionable, as follows:

1. The syllable is the graphic unit, written within the confines of an approximate square.
2. The first sign of any syllable must be a consonant.[5] If the initial phoneme is vocalic, the consonant ○ is arbitrarily prefixed to it. The graphic unit is thus minimally composed of at least two signs.
3. The second sign of any graphic unit must be a vowel.
4. If the second (vowel) sign is 'vertical' it is written *to the right* of the first (consonantal) sign.
5. If the second (vowel) sign is 'horizontal' it is written *below* the first (consonantal) sign.
6. If the second sign is a 'horizontal' vowel, the third sign may be a 'vertical' vowel, forming a diphthong.[6] Such a 'vertical' vowel is written *to the right* of the preceding *two* signs.
7. The sign following a single vowel or a diphthong must be a consonant, written *below* all preceding signs.
8. An additional and final consonant may occur, written *to the right* of the immediately preceding consonant sign.

Under these rules nine possible juxtapositional arrangements are possible, as displayed in Fig. 3. The whole writing system is so simple and elegant that anyone can learn it in a matter of minutes.

Once the nature of the series and the rules of its synthesis are thus understood, an automated synthesis is achieved simply by applying the rules automatically.

In practice, the input string of *hangul* letters is first decoded to determine which of the nine possible syllabic shapes is required. Component element configurations



C = Consonant; Vv = "Vertical" vowel; Vh = "Horizontal" vowel.

Fig. 3. *Hangul* syllabic shapes.

are then called by table look-up, their sizes recalculated as required for their locations in the specified syllabic shape, and a plot configuration for the entire syllabic character is then generated.

Owing to lack of CRT output, the plot configurations are now first written on tape and later plotted on a Benson-Lehner incremental plotter, the only graphic output currently available at our facility. Provision for specifying the size of plotted characters, spacing between characters and lines and margins of plot are all built into the software, so that successive pages of text will automatically be defined from input of indeterminate size.

A synthetic approach, as described, provides the opportunity for many useful options, such as automatic addition of bold-face or skewed (italic) printout, underlining, input from any consistent romanization in addition to *hangul* itself, and the very attractive option of varying entire fonts as desired. Some samples of output using a simple and somewhat stark font (more suited to the plotter than CRT) are shown in Figs. 4-6.

Readers may be interested in seeing the results so far achieved in applying the synthetic method to Chinese. Fig. 7 is a plot of sample Chinese characters produced in a project aimed at simulating the Asian language typewriter, SINCOder (patent applied).

None of the projects mentioned have been funded, except for a small amount of support from the University of Kansas General Research Fund and the K.U. Computation Center, for which we are extremely grateful. We

[5] Doubled consonants in single sign space are treated in precisely the same manner as single consonant signs.

[6] The signs which take part in diphthong formation may be regarded as two additional subsets of the two kinds of vowel signs, consisting of the 'horizontal' vowels ㅗ, ㅜ and ㅡ, and the 'vertical' vowels ㅏ, ㅐ, ㅓ, ㅔ and ㅣ respectively. Not all possible combinations occur; those occurring are: ㅘ, ㅙ, ㅚ, ㅝ, ㅞ, ㅟ, and ㅢ. (The last is actually a vowel, Yale: OY, represented by two vowel signs in the diphthong fashion.)

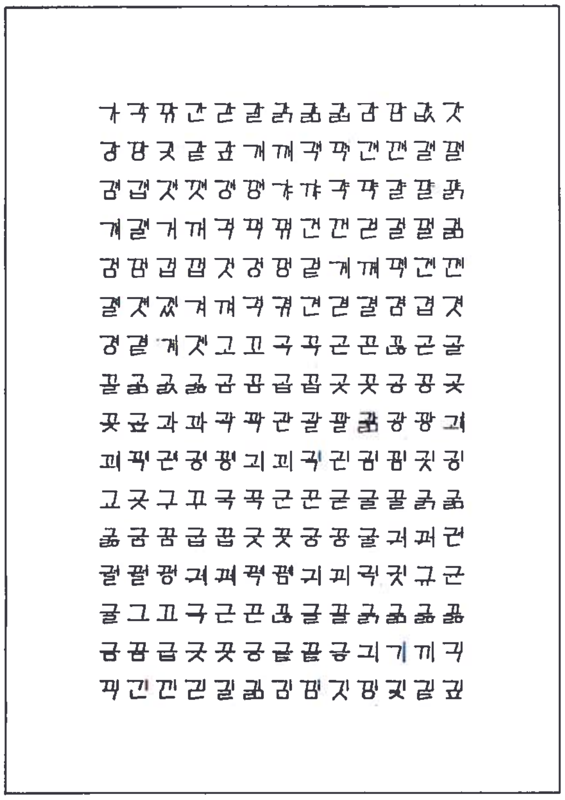
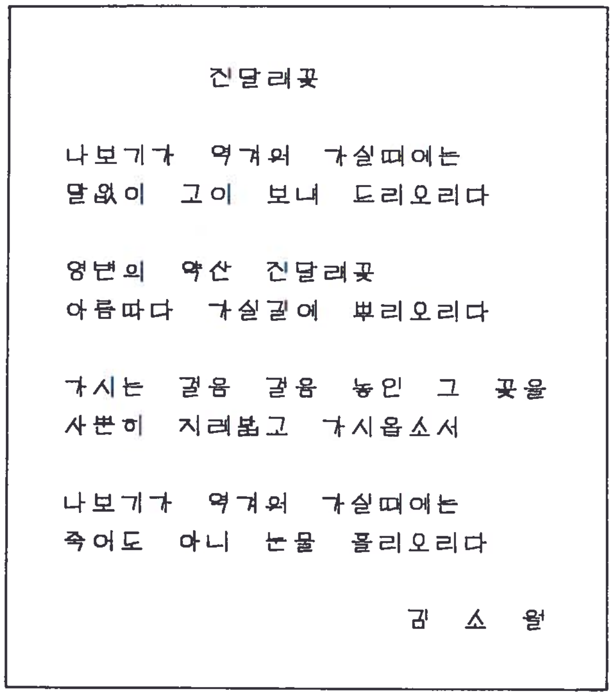


Fig. 4. A plot of syllables beginning with phoneme [k], taken from Martin, Lee and Chang, *A Korean-English Dictionary* (Yale University Press, 1967).


진달래꽃

나보기가 역겨워 가실때에는
말없이 고이 보내 드리오리다

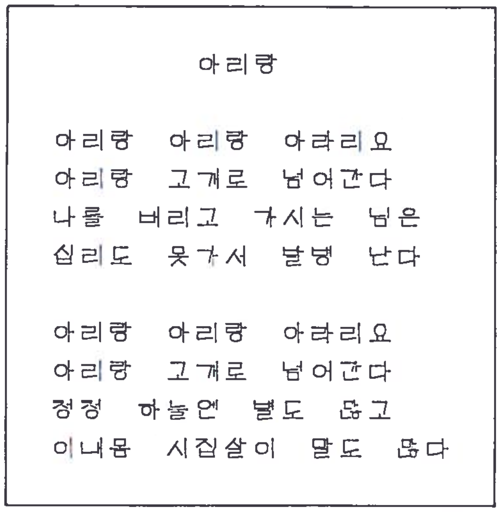
영변의 약산 진달래꽃
아름따다 가실길에 뿌리오리다

가시는 걸음 걸음 놓인 그 꽃을
사뿐히 지레밟고 가시옵소서

나보기가 역겨워 가실때에는
죽어도 아니 눈물 흘리오리다

김 소 월

Fig. 5. A plot of the famous poem "The Azalea", by Kim So-wŏl.


아리랑

아리랑 아리랑 아라리요
아리랑 고개로 넘어간다
나를 버리고 가시는 님은
십리도 못가서 발병 난다

아리랑 아리랑 아라리요
아리랑 고개로 넘어간다
정정 하늘엔 별도 많고
이내몸 시집살이 말도 많다

Fig. 6. A plot of the popular Korean folk-song "Alilang".

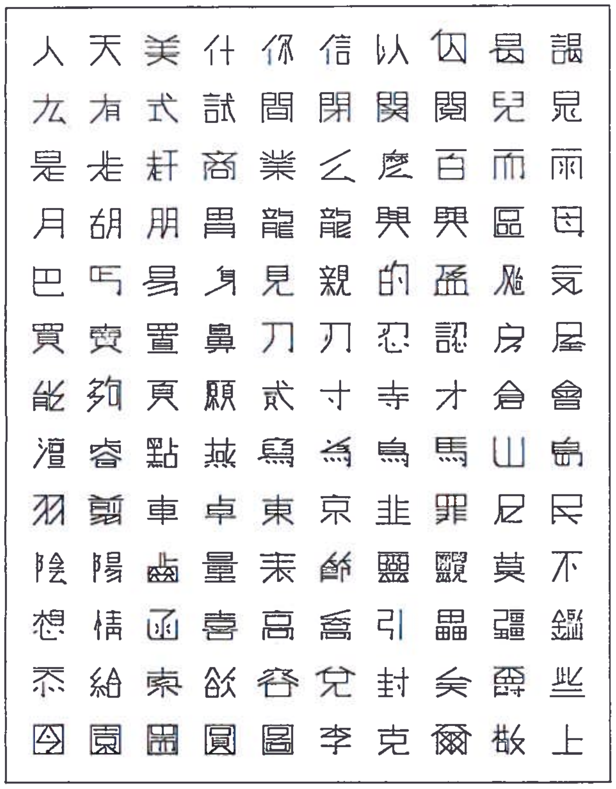


Fig. 7. SINCOder Chinese Printout.

hope that the achievement of a true synthesis for *hangul* and the well-developed state of the synthesis for Chinese will encourage funding agencies and researchers to turn their attention to the basic nature of the problem of orthographic automation. The longer the inherent imperatives of the problem are ignored in seeking ways around rather than through difficulties, the longer the advantages of automation for variable series orthographies will be delayed.

An Algorithm for Analyzing Hebrew Words

JAMES D. PRICE
Philadelphia, Pennsylvania

An algorithm is developed for analyzing the words of modern Hebrew that is based on a context-sensitive complex-constituent phrase-structure grammar of modern Hebrew morphology consisting of one rule, one initial symbol, and nine terminal symbols with fourteen grammatical attributes. It operates the rule in reverse using four sets of constraints to optimize computation. The algorithm consists of an operational function, and a set of mapping functions that reduce the analysis to simple look-up table operations. The analysis consists of inflection class, root, stem, number, gender, person, tense, mood, voice, compound-word prefixes, and pronominal suffixes. All possible analyses of ambiguous words are computed.

A computer program is given in flow-chart form with a detailed description containing eleven diagrams and fifteen look-up tables. The program employs economizing techniques that pursue only fruitful paths.

The algorithm was tested on a list of twenty-two Hebrew words. The analysis of each word was correct and exhaustive. An average of 871 computations was required for each word. Comparing this number to the twenty-two million possible computations illustrates the efficiency of the algorithm.

A sample dictionary with its associated format is provided in an appendix.

I. INTRODUCTION

This is a description of the second part of a research program on machine translation from Hebrew to English. The first part is described in an article entitled "An Algorithm for Generating Hebrew Words",* in which a procedure is developed that produces the correct orthography of an inflected Hebrew word from a complete grammatical description of the word.

This article describes an algorithm for analyzing Hebrew words that essentially consists of operating the rules of the generating algorithm in reverse. In addition, the algorithm takes into account the problems of compound words, and techniques for optimizing computing time. The algorithm begins with an inflected Hebrew word in transliterated orthography, and computes one or more complete grammatical descriptions of the word. The description consists of the root, stem, number, gender person, and all other grammatical attributes of the word. Justification of the linguistic data presented is beyond the scope of this article.

There are a few differences between the analysis algorithm and the generating algorithm: (1) In the analysis algorithm, word boundaries are not used. (2) The generating algorithm is free of ambiguity; the analysis algorithm computes every identity of an ambiguous word. (3) The generating algorithm is not concerned with compound words but the analysis algorithm separates the elements of a compound word and analyzes each.

This article is written with the assumption that the reader is familiar with the previous work. Many of the terms and symbols used here are identical with those previously used. Such terms and symbols will not be fully defined again. These include the table of transliteration, the table of root classes, the table of pronominal suffix classes, the table of "logeme" classes. The identity of symbols is given in the list of symbols.

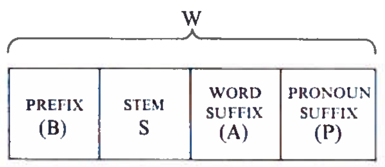
II. A STRUCTURAL MODEL OF HEBREW WORDS

The structural model developed in the previous article was adapted to the word generating algorithm. Figure 1A illustrates this model, the structure of which is represented by the expression

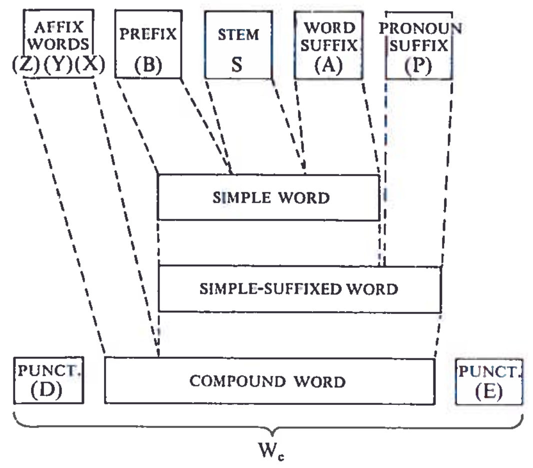
$$W = L\ (B)\ S\ (A)\ (P)\ R \qquad (1)$$

* Published in *Computer Studies in the Humanities and Verbal Behavior*, Vol. 1, No. 2.

where the letters represent the elements of a word and the parentheses enclose optional elements.



(A) Structure of a Hebrew Word



(B) Structure of a Compound Hebrew Word

Fig. 1. Structural Model of Hebrew Words.

For the analysis of words, a more complex model is required because of compound words. Figure 1B illustrates the model of a compound word the structure of which is represented by the expression

$$C = (D)\,(Z)\,(Y)\,(X)\,W\,(E) \qquad (2)$$

where the letters represent the elements of a compound word and the parentheses enclose optional elements. In this model, the central word (W) is represented by the expression

$$W = (B)\,S\,(A)\,(P) \qquad (3)$$

which differs from the generating model by the absence of word boundaries (L, R). This model indicates that a compound Hebrew word may consist of as many as seven connected simple words, two of which are punctuation marks (D, E), and one of which is a pronominal suffix (P). An example of such a compound word is

"WLKSYWLYKWNW, ("and when they brought us,)

This example contains all but one element (X); it breaks down as follows:

Symbol:	D	Z	Y	B	S	A	P	E
Orthographic Value:	"	W	LKS	Y	WLYK	W	NW	,

The affixed words (X, Y, Z) are three classes of Hebrew words that always are prefixed to the following word with no intervening space; they are called inseparable prefixes. Class 1 affixed words (X) may appear before simple words only; Class 2 affixed words (Y) may appear before simple words or Class 1 affixed words; Class 3 affixed words (Z) may appear before simple words, Class 2 or Class 1 affixed words. Table 1 and 2 list the members of word constituents D and E respectively; Tables 3, 4, and 5 list the members of word constituents X, Y, and Z respectively.

TABLE 1
Word Constituent D_k

	(1)	(2)
k	D	u
1	*	0
2	"	1
0	(	2

Expressions (2) and (3) may be combined and expanded into the expression

$$C_{abcdeghjuv1234} = D_u Z_j Y_{adh} X_{adg} B_{ad} S_{abcde1234} A_{ade} P_{ade} E_v \qquad (4)$$

where the subscripts represent the grammatical attributes of the compound word (C) the definitions of which are

List of Symbols

A	= Word Suffix
B	= Prefix
C	= Compound Word
C*	= Compound Word-Form
D	= Left Punctuation
E	= Right Punctuation
L	= Left Word-Boundary
P	= Pronominal Suffix
R	= Right Word-Boundary
S	= Stem
S*	= Stem-Form
U	= Universe
W	= Word
Y	= Affixed Word, Class 1
X	= Affixed Word, Class 2
Z	= Affixed Word, Class 3
a	= Inflection Class
b	= Stem Class
c	= Root Class
d	= Logeme Class
e	= Pronominal Suffix Class
g	= Type of Class 1 Affixed Word
h	= Type of Class 2 Affixed Word
j	= Type of Class 3 Affixed Word
k	= Index for D
l	= Type of Z
m	= Type of Y
n	= Type of X
q	= Type of B
r	= Type of S*
t	= Type of A
u	= Type of D
v	= Type of E
w	= Type of P
x	= Index for E
Δ	= Hebrew Dictionary
Φ	= The set of Hebrew characters
ψ	= Mapping Function
ω	= Analysis Index
Ω	= Operational Function
1	= First Root Letter
2	= Second Root Letter
3	= Third Root Letter
4	= Fourth Root Letter
∈	= "is an element of"
{ }	= Set
∧	= Logical "and"

contained in the list of symbols. The values of the subscripts govern the orthographic value of the associated symbol. The values for constituents D, E, X, Y, and Z are computed from Tables 1 through 5 respectively. The

TABLE 2
Word Constituent E_x

	(1)	(2)
x	E	v
0	*	0
1	,	1
2	.	2
3	;	3
4	!	4
5	"	5
6	?	6
7	)	7
8	-	8

TABLE 3
Word Constituent X_n

	(1)	(2)	(3)	(4)	(5)
n	x	a	d	g	#
1	*	1	>0	0	0
2	H	2	10-18	1	1
3	*	2, 3	>0	0	0
4	H	3	>0	1	1

TABLE 4
Word Constituent Y_m

	(1)	(2)	(3)	(4)	(5)	Meaning
m	Y	a	d	h	#	
1	LKS	1	20-29	1	3	when
2	MS	1	>0	2	2	when
3	*	1	>0	0	0	(zero)
4	H	1	>0	7	1	(sign of ?)
5	S	1	>0	8	1	who/which
6	BS	1	>0	9	2	because
7	KS	1	>0	10	2	as/when
8	K	2	10-18	3	1	like/as
9	L	2	10-18	4	1	to/for
10	B	2	10-18	5	1	in
11	M	2	10-18	6	1	from
12	*	2, 3	>0	0	0	(zero)
13	H	2, 3	>0	7	1	(sign of ?)
14	S	2, 3	>0	8	1	who/which
15	BS	2, 3	>0	9	2	because
16	KS	2, 3	>0	10	2	as/when
17	K	3	>0	3	1	like/as
18	L	3	>0	4	1	to/for
19	B	3	>0	5	1	in
20	M	3	>0	6	1	from

values of constituents B, S, A, and P are computed from Tables listed in the previous article which are not repeated here; these data appear later in another form in Tables 7 through 10 respectively. Expression (4) is the complete structural model of a Hebrew word.

TABLE 5
Word Constituent Z_l

	(1)	(2)	(3)	Meaning
l	z	j	#	
1	*	0	0	(zero)
2	W	1	1	and

III. THE ALGORITHM

Based on the structural model in expression (4), the algorithm for analyzing Hebrew words consists of an operational function (Ω) and a small set of mapping functions (Ψ). The input of the algorithm is a Hebrew word (C) of unknown parsing, consisting of a sequence of transliterated Hebrew characters bounded on each side by a space. The operational function (Ω) divides the word (C) into its constituent members D, Z, Y, X, B, S, A, P, E. Often a word can be divided into more than one set of constituent members. Therefore, the operational function (Ω) is required to produce all possible sets of constituent members. This is expressed by

$$(C) = \begin{Bmatrix} D_{k_1}\ Z_{l_1}\ Y_{m_1}\ X_{n_1}\ B_{q_1}\ S_{r_1}\ A_t\ P_{w_1}\ E_{x_1} \\ D_{k_2}\ Z_{l_2}\ Y_{m_2}\ X_{n_2}\ B_{q_2}\ S_r\ A_{t_2}\ P_{w_2}\ E_{x_2} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ D_{k_\omega}\ Z_{l_\omega}\ Y_{m_\omega}\ X_{n_\omega}\ B_{q_\omega}\ S_{r_\omega}\ A_{t_\omega}\ P_{w_\omega}\ E_{x_\omega} \end{Bmatrix} \qquad (5)$$

where the subscripts 1, 2, ... ω are indices only, and do not correspond to the attribute subscripts of expression (4).

Each set of constituent members of expression (5) represents an analysis of the input word.

For each set of constituent members of expression (5) a set of values may be computed for the attribute subscripts (a, b, c, d, e, g, h, j, u, v, 1, 2, 3, 4) of the symbols, the values of which are a complete grammatical description of the given word. The values of these subscripts are computed by the mapping functions (Ψ). A given set of constituent members may have more than one parsing, i.e., more than one set of attribute subscripts. For example, the Hebrew word, Q@LT, may be divided into the constituents S = Q@L (kill), A = T, all others being zero value. This word is parsed as:

(1) verb, past tense, second person, feminine, singular;
(2) verb, past tense, second person, masculine, singular;
(3) verb, past tense, third person, feminine, singular.

The operational function (Ω) produces an output for each possibility in ambiguous cases.

The next sections define the manner in which the operational function (Ω) and the mapping functions (Ψ) operate.

A. *The Mapping Functions* (Ψ)

Mapping functions are look-up tables that define the value of a dependent variable for given values of an independent variable. To illustrate, consider Table 6 (Ψ_6), with independent variable x, and dependent variables u and v.

TABLE 6

x	u	v
1	2	A
2	4	B
3	6	C
4	8	D

The value in vertical column u and horizontal row x of Table 6 is designated by the notation

$$\Psi_6\,(u, x)$$

Likewise, the value in column v and row x of Table 6 is designated by

$$\Psi_6\,(v, x)$$

then if

$$y = \Psi_6\,(u, x)$$

The value of y is 6 when x = 3, and y = 2 when x = 1, etc. Further if

$$z = \Psi_6\,(v, x)$$

z has the orthographic value *B* when x = 2 etc. The above illustrates *fixed* mapping functions.

There is another type of mapping function called herein a variable mapping function. The *variable* mapping function does not have a single table to define its values; instead, it constructs its own look-up table from specified lists. The variable mapping function is designated by the following notation

$$\Psi_{yx}$$

where y designates a list of values for the dependent variable, and x designates a list of values for the independent variable. For example, given the lists

$$\Lambda_1 = 1, 2, 3, 4, 5$$
$$\Lambda_2 = H, B, D, Y, L$$
$$\Lambda_3 = H, 1, 2, Y, 3$$

a table can be constructed from the lists Λ_1 and Λ_2 as follows

$$\Psi_{\Lambda_2\,\Lambda_1}$$

Λ_1	Λ_2
1	H
2	B
3	D
4	Y
5	L

The variable mapping function is designated by

$$\Psi_{\Lambda_2\,\Lambda_1}$$

A specific value of the function (say $\Lambda_2 = 2$) is designated by

$$\Psi_{\Lambda_2\,\Lambda_1}(2) = B$$

Given the same set of lists a new table ($\Psi_{\Lambda_2\,\Lambda_3}$) can be constructed from which the following specific values are obtained

$$\Psi_{\Lambda_2\,\Lambda_3}(1) = B$$
$$\Psi_{\Lambda_2\,\Lambda_3}(2) = D$$
$$\Psi_{\Lambda_2\,\Lambda_3}(3) = L$$

The dependent variable has zero value (*) for undefined values of the independent variable. Thus

$$\Psi_{\Lambda_2\,\Lambda_3}(4) = *$$

The above illustrates *variable* mapping functions.

B. *The Operational Function* (Ω)

This section describes the computations of the operational function (Ω) in formalized terms. The resultant number of computations will appear to be large. However, in programming the function, several optimizing techniques are used to reduce the actual number of computations to a small percentage of the total indicated.

Referring back to expression (4), the structure of a compound Hebrew word (C) is expressed in terms of its constituent members (D, Z, Y, X, B, S, A, P, E). Each symbol (except S) represents a small set of orthographic values that may occupy the structural position of the symbol in a given word. As such, these word constituents readily are identified by simple comparing techniques. However, the symbol S represents the stem of the word which always contains arbitrary characters depending on the root of the word. To recognize the stem, a technique is used for identifying the *form* of the stem and for isolating the root letters. This is accomplished by using a set of *stem forms* which are compared with the characters of the input word. For example, the form of the *Hiphil* Stem is represented by

H 1 2 Y 3

where the H and Y are fixed orthographic values, and the numerals 1, 2, and 3 occupy the position of the first, second, and third root letter respectively. A prospective stem is compared to the *stem form* according to the criteria: (1) Does the prospective stem and the *stem form* have the same number of characters? (2) Are the alphabetic characters of the *stem form* the same as the corresponding characters of the prospective stem? If the answer to both questions is yes, the prospective stem is said to properly correspond to the *stem form*. To illustrate, let the prospective stem HBDYL ('to divide', root BDL) be compared to the above *stem form* as follows

H B D Y L
H 1 2 Y 3

The requirements of the criteria are met, and further it is seen that the numerals, 1, 2, and 3 mark the position of the root letters BDL.

Now, let the symbol S* represent a *stem form* rather than a stem which has the following subscript descriptors as compared to a stem S:

$$S^*_{abcde} \qquad (6)$$
$$S_{abcde1234}$$

It is observed that the only difference between the two is that the stem S has root letters specified in the subscripts 1, 2, 3, and 4.

The symbols (D, Z, Y, X, B, A, P, E) each represent a small set of orthographic values, and the symbol S* represents a larger set of stem forms the members of which are listed in tables as follows:

Symbol	D	Z	Y	X	B	S*	A	P	E
Table	1	5	4	3	7	10	8	9	2

Let {D} represent all members of a set of orthographic values of D, and let D_k represent the kth member of that set. Likewise, let {Y} represent all the members of a set of orthographic values of Y, and let Y_m represent the mth member of that set, and so forth for all the symbols.

TABLE 7
Word Constituent B_q

	(1)	(2)	(3)	(4)
q	B	a	d	#
1	*	1	10-19 32-38	0
2	Y	1	20, 25	1
3	T	1	21-23 26-28	1
4	A	1	24	1
5	N	1	29	1
6	*	2, 3	>0	0

Then the structure of a given *wordform* (C*) is represented by the expression

$$C^*_{klmnqrtwx} = D_k \, Z_l \, Y_m \, X_n \, B_q \, S^*_r \, A_t \, P_w \, E_x \qquad (7)$$

where

$$D_k = \Psi_1 (1, k) \qquad (8)$$
$$Z_l = \Psi_5 (1, l) \qquad (9)$$
$$Y_m = \Psi_4 (1, m) \qquad (10)$$
$$X_n = \Psi_3 (1, n) \qquad (11)$$
$$B_q = \Psi_7 (1, q) \qquad (12)$$
$$S^*_r = \Psi_{10} (1, r) \qquad (13)$$
$$A_t = \Psi_8 (1, t) \qquad (14)$$
$$P_w = \Psi_9 (1, w) \qquad (15)$$
$$E_x = \Psi_2 (1, x) \qquad (16)$$

The term *wordform* is used here rather than *word* because the symbol, S*, is a *stem form* rather than a specific *stem*. The result is a general word form rather than a specific word.

The universe of all possible *word forms* consists of all possible combinations of the symbols of expression (7). This is expressed as

$$U = C^*_{klmnqrtwx} \qquad (17)$$

The task of the Operation Function (Ω) is to select from this universe the word forms that correspond to the input word. This selection process consists of choosing from the universe (U) those word forms that meet the following conditions: (1) word-structure conditions, (2) grammatical conditions, (3) root-structure conditions, (4) lexical conditions. The first condition requires the system to consider only word forms that match the word-structure of the input word. The second condition is related to the grammatical structure of the language: it requires the system to consider only grammatical combinations of word constituents. The third condition requires the system to consider only word forms that match the root-structure of the input word. The fourth condition requires

TABLE 8
Word Constituent A_t

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
t	A	a	d	SE	#	q_{min}	q_{max}	n_{min}	n_{max}	m_{min}	m_{max}
1	T	1	11	1	1	1	1	1	1	2	7
2	TY	1	13	1	2	1	1	1	1	2	7
3	TW	1	17	1	2	1	1	1	1	2	7
4	TW	1	18	1	2	1	1	1	1	2	7
5	W	1	26	1	1	3	3	1	1	1	7
6	W	1	28	1	1	3	3	1	1	1	7
7	W	1	38	1	1	1	1	1	1	2	7
8	*	1	10	2	0	1	1	1	1	2	7
9	T	1	12	2	1	1	1	1	1	2	7
10	TY	1	14	2	2	1	1	1	1	2	7
11	W	1	15	2	1	1	1	1	1	2	7
12	W	1	16	2	1	1	1	1	1	2	7
13	NW	1	19	2	2	1	1	1	1	2	7
14	*	1	20	2	0	2	2	1	1	1	7
15	*	1	21	2	0	3	3	1	1	1	7
16	*	1	22	2	0	3	3	1	1	1	7
17	Y	1	23	2	1	3	3	1	1	1	7
18	*	1	24	2	0	4	4	1	1	1	7
19	W	1	25	2	1	2	2	1	1	1	7
20	W	1	27	2	1	3	3	1	1	1	7
21	*	1	29	2	0	5	5	1	1	1	7
22	*	1	32	2	0	1	1	1	1	2	7
23	Y	1	33	2	1	1	1	1	1	2	7
24	W	1	37	2	1	1	1	1	1	2	7
25	H	1	11	0	1	1	1	1	1	2	7
26	T	1	13	0	1	1	1	1	1	2	7
27	TM	1	17	0	2	1	1	1	1	2	7
28	TN	1	18	0	2	1	1	1	1	2	7
29	NH	1	26	0	2	3	3	1	1	1	7
30	NH	1	28	0	2	3	3	1	1	1	7
31	NH	1	38	0	2	1	1	1	1	1	7
32	*	2	10	0	0	6	6	2	3	8	16
33	T	2	11	0	1	6	6	2	3	8	16
34	H	2	11	0	1	6	6	2	3	8	16
35	YM	2	15	0	2	6	6	2	3	8	16
36	WT	2	16	0	2	6	6	2	3	8	16
37	*	2	20	2	0	6	6	3	3	12	16
38	*	3	10	0	0	6	6	3	4	12	20
39	H	3	11	0	1	6	6	3	4	12	20
40	YYM	3	14	0	3	6	6	3	4	12	20
41	YM	3	15	0	2	6	6	3	4	12	20
42	WT	3	16	0	2	6	6	3	4	12	20
43	*	2	12	2	0	6	6	2	3	8	16
44	T	2	13	2	1	6	6	2	3	8	16
45	Y	2	17	2	1	6	6	2	3	8	16
46	WT	2	18	2	2	6	6	2	3	8	16
47	*	2	21	2	0	6	6	3	3	12	16
48	*	3	12	2	0	6	6	3	4	12	20
49	T	3	13	2	1	6	6	3	4	12	20
50	Y	3	17	2	1	6	6	3	4	12	20
51	WT	3	18	2	2	6	6	3	4	12	20

TABLE 9

Word Constituent P_w

	(1)	(2)	(3)	(4)	(5)	(6)	(7)
w	P	a	SD	e	#	t_{min}	t_{max}
1	*	>0	00	0	0	8	52
2	H	1	60	31	1	1	24
3	K	1	54	20	1	1	24
4	K	1	56	21	1	1	24
5	M	1	60	35	1	1	24
6	N	1	60	36	1	1	24
7	W	1	57	30	1	1	24
8	HW	1	60	30	2	1	24
9	KM	1	54	25	2	1	24
10	KN	1	54	26	2	1	24
11	NW	1	67	15	2	1	24
12	NY	1	67	10	2	1	24
13	YH	1	59	31	2	1	24
14	YK	1	55	21	2	1	24
15	YM	1	59	35	2	1	24
16	YN	1	59	36	2	1	24
17	YHW	1	59	30	3	1	24
18	YNW	1	58	15	3	1	24
19	YNY	1	58	10	3	1	24
20	H	2, 3	63	31	1	43	51
21	K	2, 3	63	20	1	43	51
22	M	2, 3	64	35	1	43	51
23	N	2, 3	64	36	1	43	51
24	W	2, 3	63	30	1	43	51
25	Y	2, 3	61	10	1	43	51
26	Y	2, 3	65	40	1	43	51
27	HM	2, 3	65	35	2	43	51
28	HN	2, 3	65	36	2	43	51
29	KM	2, 3	63	25	2	43	51
30	KN	2, 3	63	26	2	43	51
31	NW	2, 3	65	15	2	43	51
32	YH	2, 3	66	31	2	43	51
33	YK	2, 3	66	20	2	43	51
34	YK	2, 3	63	21	2	43	51
35	YW	2, 3	66	30	2	43	51
36	YHM	2, 3	66	35	3	43	51
37	YHN	2, 3	66	36	3	43	51
38	YKM	2, 3	66	25	3	43	51
39	YKN	2, 3	66	26	3	43	51
40	YNW	2, 3	62	15	3	43	51
41	YNY	2, 3	62	40	3	43	51
42	YYK	2, 3	66	21	3	43	51
43	YYK	2, 3	72	20	3	43	51

the system to consider only those parsings of input word that are in the dictionary. These four conditions are defined and formalized in the material that follows.

(1) *Word-Structure Conditions*

The Operation Function (Ω) considers only those word forms that correspond to the input word in the following manner:

(1) the word form has the same number of characters as the input word,

TABLE 10

*Word Constituent S**

	(1)	(2)	(3)	(4)	(5)	(6)
r	S*	a	b	SC	SD	SE
1	2	1	07	29	08	02
2	2	1	07	29	27	01
3	2	2	13	27	72	01
4	1	3	26	00	49	02
5	12	1	07	06	08	00
6	12	1	11	30	75	00
7	12	1	12	30	75	00
8	12	1	14	29	75	00
9	12	1	14	05	75	00
10	23	1	07	29	27	00
11	12	1	11	03	07	02
12	12	1	11	03	14	02
13	12	1	11	30	33	02
14	12	1	12	30	11	02
15	12	1	12	03	30	02
16	12	1	12	03	14	02
17	12	1	12	03	09	02
18	12	1	13	03	07	02
19	12	1	13	03	14	02
20	12	1	14	03	14	02
21	12	1	14	29	11	02
22	12	1	14	05	24	02
23	12	1	04	07	32	02
24	12	1	12	27	12	02
25	W2	1	08	29	08	02
26	Y2	1	14	29	08	02
27	13	1	11	02	06	02
28	13	1	12	02	12	02
29	13	1	13	02	12	02
30	13	1	14	02	05	02
31	13	1	14	04	22	02
32	23	1	11	26	04	02
33	23	1	12	26	04	02
34	23	1	13	26	04	02
35	23	1	12	27	04	02
36	2Y	1	07	29	09	02
37	12	1	11	30	10	01
38	12	1	11	30	13	01
39	12	1	12	30	13	01
40	12	1	14	29	13	01
41	12	1	14	05	26	01
42	12	1	04	07	27	01
43	W2	1	12	30	73	01
44	Y2	1	11	30	73	01
45	M2	2	07	29	77	00
46	13	2	12	02	14	00
47	13	2	12	04	14	00
48	13	2	13	02	14	00
49	13	2	13	04	14	00
50	13	2	14	02	14	00
51	13	2	14	04	14	00
52	2T	2	13	27	72	00
53	M2	2	07	29	42	02
54	13	2	12	02	39	02
55	13	2	12	04	39	02
56	13	2	13	02	39	02
57	13	2	13	04	39	02
58	13	2	14	02	39	02
59	13	2	14	04	39	02
60	12	3	30	00	52	00
61	12	3	28	00	77	00
62	12	3	27	00	76	00

Table 10 — *Continued*

r	(1) S*	(2) a	(3) b	(4) SC	(5) SD	(6) SE
63	12	3	25	00	76	00
64	13	3	30	00	51	00
65	14	3	28	00	68	00
66	14	3	29	00	78	00
67	12	3	27	00	42	02
68	12	3	28	00	47	02
69	12	3	25	00	39	02
70	12	3	30	00	50	02
71	13	3	30	00	49	02
72	14	3	29	00	50	02
73	123	1	11	30	10	00
74	123	1	11	30	13	00
75	123	1	12	30	13	00
76	123	1	14	00	17	00
77	123	1	14	26	19	00
78	123	1	14	29	13	00
79	123	1	14	05	26	00
80	123	1	04	07	27	00
81	W23	1	12	30	73	00
82	Y23	1	14	29	27	00
83	Y23	1	11	30	73	00
84	H12	1	07	06	11	02
85	N12	1	03	03	14	02
86	N12	1	03	06	28	02
87	T12	1	06	11	08	02
88	W12	1	08	06	08	02
89	W12	1	08	03	30	02
90	W12	1	12	30	08	02
91	Y12	1	03	06	08	02
92	Y12	1	03	03	30	02
93	Y12	1	03	29	08	02
94	Y12	1	14	03	08	02
95	Y12	1	03	03	73	02
96	Y12	1	11	03	73	02
97	Y12	1	12	03	73	02
98	Y12	1	13	03	73	02
99	1@2	1	06	31	08	02
100	1D2	1	06	32	08	02
101	1T2	1	06	14	08	02
102	1W2	1	11	03	15	02
103	1W2	1	12	03	15	02
104	1W2	1	13	03	15	02
105	1W2	1	14	03	27	02
106	1W2	1	14	03	32	02
107	1W2	1	05	07	33	02
108	HW2	1	08	29	28	02
109	HW2	1	08	27	12	02
110	1Y2	1	04	07	28	02
111	HY2	1	07	29	11	02
112	HY2	1	07	27	12	02
113	NY2	1	03	29	28	02
114	NY2	1	03	27	12	02
115	W13	1	08	02	03	02
116	W13	1	08	04	03	02
117	123	1	04	00	21	02
118	123	1	11	00	02	02
119	123	1	12	00	00	02
120	123	1	12	02	21	02
121	123	1	12	26	05	02
122	123	1	13	00	00	02
123	123	1	14	03	14	02
124	123	1	14	00	16	02
125	123	1	14	02	21	02

Table 10 — *Continued*

r	(1) S*	(2) a	(3) b	(4) SC	(5) SD	(6) SE
126	123	1	14	04	23	02
127	123	1	14	26	18	02
128	123	1	12	27	14	02
129	W23	1	08	16	03	02
130	W23	1	08	29	27	02
131	W23	1	12	28	73	02
132	W23	1	08	27	03	02
133	Y23	1	11	26	03	02
134	Y23	1	12	26	03	02
135	Y23	1	13	26	03	02
136	Y23	1	14	26	08	02
137	Y23	1	11	01	73	02
138	Y23	1	13	28	73	02
139	Y23	1	12	27	03	02
140	1Y3	1	07	03	30	02
141	1Y3	1	12	02	14	02
142	1Y3	1	13	02	14	02
143	1Y3	1	07	02	03	02
144	1Y3	1	07	04	03	02
145	2Y3	1	07	26	03	02
146	2Y3	1	07	27	03	02
147	12T	1	11	30	69	02
148	12T	1	12	30	69	02
149	12T	1	14	29	69	02
150	12T	1	14	05	69	02
151	12W	1	11	03	12	02
152	12W	1	12	03	12	02
153	12W	1	13	03	12	02
154	12W	1	14	03	12	02
155	12Y	1	07	06	09	02
156	12Y	1	12	30	12	02
157	12Y	1	14	29	12	02
158	12Y	1	14	05	25	02
159	12Y	1	04	07	31	02
160	W2Y	1	08	29	09	02
161	Y2Y	1	14	29	09	02
162	H12	1	07	06	13	01
163	W12	1	12	30	10	01
164	1Y2	1	04	07	68	01
165	HY2	1	07	29	13	01
166	123	1	07	06	08	01
167	Y23	1	14	26	20	01
168	12Y	1	11	30	75	01
169	12Y	1	12	30	75	01
170	12Y	1	14	29	75	01
171	12Y	1	14	05	75	01
172	Y2Y	1	14	29	27	01
173	M12	2	07	06	77	00
174	M12	2	04	07	77	00
175	N12	2	03	06	77	00
176	N12	2	03	03	14	00
177	1W2	2	12	03	77	00
178	1W2	2	14	07	77	00
179	MW2	2	08	29	77	00
180	NY2	2	03	29	77	00
181	123	2	04	00	71	00
182	123	2	12	03	71	00
183	123	2	02	02	14	00
184	123	2	13	00	14	00
185	123	2	04	07	71	00
186	123	2	14	07	71	00
187	123	2	12	02	71	00
188	123	2	12	04	71	00

TABLE 10 — *Continued*

	(1)	(2)	(3)	(4)	(5)	(6)
r	S*	a	b	SC	SD	SE
189	123	2	13	02	71	00
190	123	2	13	04	71	00
191	123	2	14	02	71	00
192	123	2	14	04	71	00
193	H23	2	07	29	71	00
194	M23	2	07	29	41	00
195	M12	2	07	06	42	02
196	M12	2	04	07	42	02
197	N12	2	03	03	39	02
198	N12	2	03	06	42	02
199	1W2	2	12	03	42	02
200	1W2	2	14	03	72	02
201	1W2	2	14	07	42	02
202	MW2	2	08	29	42	02
203	NY2	2	03	29	42	02
204	123	2	02	02	39	02
205	123	2	04	00	40	02
206	123	2	13	00	43	02
207	123	2	12	02	40	02
208	123	2	12	04	40	02
209	123	2	13	02	40	02
210	123	2	13	04	40	02
211	123	2	14	02	40	02
212	123	2	14	04	40	02
213	23T	2	13	26	72	02
214	M12	2	07	06	70	01
215	M12	2	04	07	70	01
216	1W2	2	12	03	70	01
217	1W2	2	14	07	41	01
218	123	3	28	00	44	00
219	123	3	01	00	76	00
220	123	3	01	00	39	02
221	123	3	28	00	66	02
222	124	3	29	00	49	02
223	12Y	3	27	00	70	02
224	H123	1	07	06	13	00
225	W123	1	12	30	10	00
226	1Y23	1	04	07	68	00
227	HY23	1	07	29	13	00
228	Y2W3	1	14	26	20	00
229	HT12	1	06	11	11	02
230	HW12	1	08	03	14	02
231	HW12	1	08	06	28	02
232	HY12	1	03	03	15	02
233	HY12	1	03	06	29	02
234	H1@2	1	06	31	11	02
235	H1D2	1	06	32	11	02
236	H1T2	1	06	14	11	02
237	Y1W2	1	14	03	20	02
238	HW13	1	08	02	35	02
239	HW13	1	08	04	35	02
240	H123	1	07	00	12	02
241	N123	1	03	00	05	02
242	N123	1	03	06	68	02
243	T123	1	06	00	03	02
244	T123	1	06	11	27	02
245	W123	1	08	00	03	02
246	W123	1	12	28	07	02
247	Y123	1	03	00	03	02
248	Y123	1	03	06	27	02
249	Y123	1	03	29	27	02
250	Y123	1	11	02	03	02
251	1@23	1	06	24	03	02

TABLE 10 — *Continued*

	(1)	(2)	(3)	(4)	(5)	(6)
r	S*	a	b	SC	SD	SE
252	1@23	1	06	31	27	02
253	1@23	1	06	35	03	02
254	1D23	1	06	32	27	02
255	1D23	1	06	36	03	02
256	1D23	1	06	25	03	02
257	1T23	1	06	10	03	02
258	1T23	1	06	14	27	02
259	1T23	1	06	33	03	02
260	1T23	1	06	34	03	02
261	1W23	1	05	00	00	02
262	1W23	1	04	02	21	02
263	1W23	1	09	17	00	02
264	1W33	1	09	04	00	02
265	HW23	1	08	16	35	02
266	HW23	1	08	29	68	02
267	HW23	1	08	27	14	02
268	1Y23	1	04	04	00	02
269	1Y23	1	04	02	05	02
270	1Y23	1	04	00	05	02
271	HY23	1	07	26	12	02
272	NY23	1	03	26	05	02
273	NY23	1	03	29	68	02
274	12W3	1	11	00	01	02
275	H1Y3	1	07	03	38	02
276	H1Y3	1	07	02	04	02
277	H1Y3	1	07	04	04	02
278	12Y3	1	07	00	03	02
279	H2Y3	1	07	22	04	02
280	1234	1	04	43	00	02
281	H12T	1	07	06	69	02
282	N12T	1	03	06	69	02
283	1W2T	1	05	07	69	02
284	HW2T	1	08	29	69	02
285	1Y2T	1	04	07	69	02
286	HY2T	1	07	29	69	02
287	NY2T	1	03	29	69	02
288	N12W	1	03	03	12	02
289	H12Y	1	07	06	36	02
290	N12Y	1	03	06	12	02
291	T12Y	1	06	11	09	02
292	W12Y	1	08	06	09	02
293	W12Y	1	08	06	09	02
294	W12Y	1	12	30	09	02
295	Y12Y	1	03	03	09	02
296	Y12Y	1	03	06	09	02
297	Y12Y	1	03	29	09	02
298	1@2Y	1	06	31	09	02
299	1D2Y	1	06	32	09	02
300	1T2Y	1	06	14	09	02
301	1W2Y	1	14	03	31	02
302	1W2Y	1	11	03	75	02
303	1W2Y	1	12	03	75	02
304	1W2Y	1	13	03	75	02
305	1W2Y	1	05	07	34	02
306	HW2Y	1	08	29	12	02
307	1Y2Y	1	04	07	12	02
308	HY2Y	1	07	29	36	02
309	NY2Y	1	03	29	12	02
310	1Y3Y	1	07	03	09	02
311	12W3	1	14	00	17	01
312	12W3	1	14	26	19	01
313	MT12	2	06	11	77	00
314	MW12	2	08	06	77	00

Table 10 — *Continued*

	(1)	(2)	(3)	(4)	(5)	(6)
r	S*	a	b	SC	SD	SE
315	M1@2	2	06	31	77	00
316	M1D2	2	06	32	77	00
317	M1T2	2	06	14	77	00
318	M1W2	2	05	07	77	00
319	MW13	2	08	17	14	00
320	MW13	2	08	04	14	00
321	H123	2	07	06	71	00
322	M123	2	07	06	41	00
323	M123	2	04	00	14	00
324	M123	2	04	07	41	00
325	N123	2	03	00	14	00
326	1W23	2	12	03	41	00
327	1W23	2	04	02	71	00
328	1W23	2	13	26	14	00
329	1W23	2	14	00	14	00
330	1W23	2	14	07	41	00
331	MW23	2	08	23	14	00
332	1Y23	2	04	04	71	00
333	NY23	2	03	29	68	00
334	NY23	2	03	22	14	00
335	1W33	2	09	17	71	00
336	1W33	2	09	04	71	00
337	12W3	2	12	00	71	00
338	12W3	2	14	00	71	00
339	12W3	2	13	00	71	00
340	12W3	2	13	26	71	00
341	12W3	2	02	00	14	00
342	H1Y3	2	07	17	71	00
343	H1Y3	2	07	04	71	00
344	M1Y3	2	07	17	14	00
345	M1Y3	2	07	04	14	00
346	H2Y3	2	07	22	71	00
347	M2Y3	2	07	22	14	00
348	MT12	2	06	11	42	02
349	MW12	2	08	06	42	02
350	HY12	2	03	03	72	02
351	M1@2	2	06	31	42	02
352	M1D2	2	06	32	42	02
353	M1T2	2	06	14	42	02
354	M1W2	2	05	07	42	02
355	MW13	2	08	17	39	02
356	MW13	2	08	04	39	02
357	M123	2	04	00	39	02
358	N123	2	03	00	39	02
359	1W23	2	13	26	39	02
360	1W23	2	14	00	39	02
361	1W23	2	04	02	40	02
362	MW23	2	08	23	39	02
363	1Y23	2	04	04	40	02
364	NY23	2	03	22	39	02
365	NY23	2	03	29	70	02
366	1W33	2	09	17	40	02
367	1W33	2	09	04	40	02
368	12W3	2	02	00	39	02
369	12W3	2	12	00	40	02
370	12W3	2	14	00	40	02
371	12W3	2	11	00	00	02
372	H1Y3	2	07	17	40	02
373	H1Y3	2	07	04	40	02
374	M1Y3	2	07	17	39	02
375	M1Y3	2	07	04	39	02
376	H2Y3	2	07	22	40	02
377	M2Y3	2	07	22	39	02
378	12WT	2	12	03	72	02
379	12WT	2	04	07	72	02
380	12WT	2	14	07	72	02
381	H2WT	2	07	29	72	02
382	A123	3	07	00	76	00
383	H123	3	09	00	76	00
384	M123	3	12	00	76	00
385	N123	3	18	00	76	00
386	S123	3	21	00	76	00
387	T123	3	15	00	76	00
388	1W23	3	04	00	76	00
389	1Y23	3	05	00	76	00
390	12W3	3	03	00	76	00
391	12Y3	3	02	00	76	00
392	1234	3	29	00	48	00
393	1234	3	19	00	76	00
394	123N	3	22	00	76	00
395	123Y	3	24	00	76	00
396	A123	3	07	00	39	02
397	H123	3	09	00	39	02
398	M123	3	12	00	39	02
399	N123	3	18	00	39	02
400	S123	3	21	00	39	02
401	T123	3	15	00	39	02
402	1W23	3	04	00	39	02
403	1Y23	3	05	00	39	02
404	12W3	3	03	00	39	02
405	12Y3	3	02	00	39	02
406	1234	3	19	00	39	02
407	123N	3	22	00	39	02
408	123Y	3	24	00	39	02
409	HT123	1	06	11	13	02
410	HT123	1	06	00	35	02
411	HW123	1	08	00	35	02
412	HY123	1	03	00	04	02
413	HY123	1	03	29	04	02
414	HY123	1	03	06	74	02
415	H1@23	1	06	24	35	02
416	H1@23	1	06	31	13	02
417	H1@23	1	06	35	35	02
418	H1D23	1	06	32	13	02
419	H1D23	1	06	36	35	02
420	H1D23	1	06	25	35	02
421	H1T23	1	06	14	13	02
422	H1T23	1	06	10	35	02
423	H1T23	1	06	33	35	02
424	H1T23	1	06	34	35	02
425	T1W23	1	10	18	03	02
426	T1W23	1	06	37	03	02
427	1@W23	1	06	38	03	02
428	1@W23	1	10	20	03	02
429	1DW23	1	06	39	03	02
430	1DW23	1	10	21	03	02
431	1TW23	1	06	12	03	02
432	1TW23	1	10	19	03	02
433	T1Y23	1	06	40	03	02
434	1@Y23	1	06	41	03	02
435	1DY23	1	06	42	03	02
436	1TY23	1	06	13	03	02
437	HY1Y3	1	07	02	14	02
438	HY1Y3	1	07	04	14	02
439	H12Y3	1	07	00	37	02
440	HY2Y3	1	07	26	14	02

TABLE 10 — *Continued*

r	(1) S*	(2) a	(3) b	(4) SC	(5) SD	(6) SE
441	HY2Y3	1	07	27	14	02
442	T1234	1	06	43	03	02
443	1W234	1	05	43	00	02
444	HT12T	1	06	11	69	02
445	HW12T	1	08	06	69	02
446	H1@2T	1	06	31	69	02
447	H1D2T	1	06	32	69	02
448	H1T2T	1	06	14	69	02
449	HW12W	1	08	03	12	02
450	H1Y3W	1	07	03	12	02
451	H1Y3W	1	07	02	12	02
452	H1Y3W	1	07	04	12	02
453	HT12Y	1	06	11	36	02
454	HW12Y	1	08	06	12	02
455	HY12Y	1	03	03	75	02
456	HY12Y	1	03	06	75	02
457	H1@2Y	1	06	31	36	02
458	H1D2Y	1	06	32	36	02
459	H1T2Y	1	06	14	36	02
460	H1Y3Y	1	07	03	75	02
461	HY1W2	2	03	03	71	00
462	HT123	2	06	00	71	00
463	MT123	2	06	11	68	00
464	MT123	2	06	00	14	00
465	MW123	2	08	00	14	00
466	HY123	2	03	02	71	00
467	H1@23	2	06	24	71	00
468	M1@23	2	06	31	68	00
469	M1@23	2	06	24	14	00
470	H1D23	2	06	25	71	00
471	M1D23	2	06	32	68	00
472	M1D23	2	06	25	14	00
473	H1T23	2	06	10	71	00
474	M1T23	2	06	14	68	00
475	M1T23	2	06	10	14	00
476	M1W23	2	04	02	14	00
477	M1W23	2	05	00	14	00
478	M1W23	2	09	17	14	00
479	M1W33	2	09	04	14	00
480	M1Y23	2	04	04	14	00
481	H12W3	2	03	22	71	00
482	H12W3	2	03	00	71	00
483	N12W3	2	03	22	71	00
484	N12W3	2	03	06	71	00
485	N12W3	2	03	00	71	00
486	N12W3	2	03	29	71	00
487	H12Y3	2	07	00	71	00
488	M12Y3	2	07	00	14	00
489	12WWY	2	02	07	14	00
490	HT123	2	06	00	40	02
491	HT123	2	06	11	72	02
492	MT123	2	06	00	39	02
493	MT123	2	06	11	70	02
494	MW123	2	08	00	39	02
495	HY123	2	03	02	40	02
496	HY123	2	03	22	72	02
497	HY123	2	03	00	72	02
498	H1@23	2	06	24	40	02
499	H1@23	2	06	31	72	02
500	M1@23	2	06	24	39	02
501	M1@23	2	06	31	70	02
502	H1D23	2	06	32	72	02
503	H1D23	2	06	25	40	02

TABLE 10 — *Continued*

r	(1) S*	(2) a	(3) b	(4) SC	(5) SD	(6) SE
504	M1D23	2	06	32	70	02
505	M1D23	2	06	25	39	02
506	H1T23	2	06	10	40	02
507	H1T23	2	06	14	72	02
508	M1T23	2	06	10	39	02
509	M1T23	2	06	14	70	02
510	M1W23	2	04	02	39	02
511	M1W23	2	05	00	39	02
512	M1W23	2	09	17	39	02
513	M1W23	2	09	04	39	02
514	M1Y23	2	04	04	39	02
515	H12Y3	2	07	00	40	02
516	M12Y3	2	07	00	39	02
517	H12WT	2	03	06	72	02
518	H12WT	2	07	06	72	02
519	H12WT	2	03	29	72	02
520	12WWY	2	02	07	39	02
521	A12W3	3	08	00	76	00
522	H12W3	3	11	00	76	00
523	M12W3	3	14	00	76	00
524	N12W3	3	20	00	76	00
525	T12W3	3	17	00	76	00
526	1Y2W3	3	06	00	76	00
527	H12Y3	3	10	00	76	00
528	M12Y3	3	13	00	76	00
529	T12Y3	3	16	00	76	00
530	123WN	3	23	00	76	00
531	123WT	3	32	00	45	00
532	123YT	3	31	00	45	00
533	A12W3	3	08	00	39	02
534	H12W3	3	11	00	39	02
535	M12W3	3	14	00	39	02
536	N12W3	3	20	00	39	02
537	T12W3	3	17	00	39	02
538	1Y2W3	3	06	00	39	02
539	H12Y3	3	10	00	39	02
540	M12Y3	3	13	00	39	02
541	T12Y3	3	16	00	39	02
542	123WN	3	23	00	39	02
543	123WY	3	32	00	79	02
544	123YY	3	31	00	79	02
545	123WY	3	32	00	47	01
546	123YY	3	31	00	47	01
547	HT1W23	1	10	18	05	02
548	HT1W23	1	06	37	35	02
549	H1@W23	1	06	38	35	02
550	H1@W23	1	10	20	05	02
551	H1DW23	1	06	39	35	02
552	H1DW23	1	10	21	05	02
553	H1TW23	1	06	12	35	02
554	H1TW23	1	10	19	05	02
555	HT1Y23	1	06	40	35	02
556	H1@Y23	1	06	41	35	02
557	H1DY23	1	06	42	35	02
558	H1TY23	1	06	13	35	02
559	HT1234	1	06	43	35	02
560	HT1W23	2	10	18	71	00
561	HT1W23	2	06	37	71	00
562	MT1W23	2	10	18	14	00
563	MT1W23	2	06	37	14	00
564	H1@W23	2	06	38	71	00
565	H1@W23	2	10	20	71	00
566	M1@W23	2	06	38	14	00

TABLE 10 — *Continued*

r	(1) S*	(2) a	(3) b	(4) SC	(5) SD	(6) SE
567	M1@W23	2	10	20	14	00
568	H1DW23	2	06	39	71	00
569	H1DW23	2	10	21	71	00
570	M1DW23	2	06	39	14	00
571	M1DW23	2	10	21	14	00
572	H1TW23	2	06	12	71	00
573	H1TW23	2	10	19	71	00
574	M1TW23	2	06	12	14	00
575	M1TW23	2	10	19	14	00
576	HT1Y23	2	06	40	71	00
577	MT1Y23	2	06	40	14	00
578	H1@Y23	2	06	41	71	00
579	M1@Y23	2	06	41	14	00
580	H1DY23	2	06	42	71	00
581	M1DY23	2	06	42	14	00
582	H1TY23	2	06	13	71	00
583	M1TY23	2	06	13	14	00
584	HT12WT	2	06	11	71	00
585	H1@2WT	2	06	31	71	00
586	H1D2WT	2	06	32	71	00
587	H1T2WT	2	06	14	71	00
588	HT1W23	2	10	18	40	02
589	HT1W23	2	06	37	40	02
590	MT1W23	2	10	18	39	02
591	MT1W23	2	06	37	39	02
592	H1@W23	2	06	38	40	02
593	H1@W23	2	10	20	40	02
594	M1@W23	2	06	38	39	02
595	M1@W23	2	10	20	39	02
596	H1DW23	2	06	39	40	02
597	H1DW23	2	10	21	40	02
598	M1DW23	2	06	39	39	02
599	M1DW23	2	10	21	39	02
600	H1TW23	2	06	12	40	02
601	H1TW23	2	10	19	40	02
602	M1TW23	2	06	12	39	02
603	M1TW23	2	10	19	39	02
604	HT1Y23	2	06	40	40	02
605	MT1Y23	2	06	40	39	02
606	H1@Y23	2	06	41	40	02
607	M1@Y23	2	06	41	39	02
608	H1DY23	2	06	42	40	02
609	M1DY23	2	06	42	39	02
610	H1TY23	2	06	13	40	02
611	M1TY23	2	06	13	39	02

(2) the alphabetic characters of the word form must be the same as the corresponding character in the input word.

To formalize this, let the sign ":", when it appears between two symbols such as A : B have the following meaning, "A corresponds to B in that they have the same number of characters, and that each alphabetic character of B is the same as the corresponding character in A." Now, if an input word, C, is compared to a given word form, C_i*, the above conditions are met when

$$C : C_i^* \tag{18}$$

Expression (18) is the first restraint on expression (7).

(2) *Grammatical Conditions*

Certain combinations of word constituents never occur in Hebrew due to the grammatical structure of the language. Grammatical conditions must be placed on the operational function (Ω) to prevent consideration of ungrammatical combinations. The conditions are simply this: the grammatical attributes of all word constituents must agree. A comparison of expressions (4) and (7) shows that these conditions are met when

$$\begin{aligned} a_m &= a_n = a_q = a_r = a_t = a_w \\ d_m &= d_n = d_q = d_r = d_t = d_w \\ e_r &= e_t = e_w \end{aligned} \tag{19}$$

The values of these grammatical attributes, a, d, and e, are computed by the mapping functions (Ψ) as follows:

$$a_m = \Psi_4\,(2, m) \tag{20}$$
$$d_m = \Psi_4\,(3, m) \tag{21}$$
$$a_n = \Psi_3\,(2, n) \tag{22}$$
$$d_n = \Psi_3\,(3, n) \tag{23}$$
$$a_q = \Psi_7\,(2, q) \tag{24}$$
$$d_q = \Psi_7\,(3, q) \tag{25}$$
$$a_r = \Psi_{10}\,(2, r) \tag{26}$$
$$d_r = \Psi_{10}\,(4, r) \tag{27}$$
$$e_r = \Psi_{10}\,(5, r) \tag{28}$$
$$a_t = \Psi_8\,(2, t) \tag{29}$$
$$d_t = \Psi_8\,(3, t) \tag{30}$$
$$e_t = \Psi_8\,(4, t) \tag{31}$$
$$a_w = \Psi_9\,(2, w) \tag{32}$$
$$d_w = \Psi_9\,(3, w) \tag{33}$$
$$e_w = \Psi_9\,(4, w) \tag{34}$$

It will be found that not all of these values are uniquely defined; some are defined only as a set of values due to the organization of the mapping tables. This is done to permit economical computation on the basis of classes.

The values of a_r, d_t, and e_w are uniquely defined; the others are defined as sets of values. The above conditions are met when

$$\begin{aligned} a_r &\in \{a_m\} \wedge \{a_n\} \wedge \{a_q\} \wedge \{a_t\} \wedge \{a_w\} \\ d_t &\in \{d_m\} \wedge \{d_n\} \wedge \{d_q\} \wedge \{d_r\} \wedge \{d_w\} \\ e_w &\in \{e_r\} \wedge \{e_t\} \end{aligned} \tag{35}$$

Expression (35) is the second restraint on Expression (7).

(3) *Root-Structure Conditions*

The roots of Hebrew words are divided into two categories: regular roots and irregular roots. Regular roots follow the standard pattern of inflection. Irregular roots deviate from the standard pattern under given conditions.

Certain word forms generated by the operational function (Ω) are valid for regular roots and others are valid for a given irregular root class only. The root class is designated by the value of the grammatical attribute c of expression (4). The operational function (Ω) tests the root letters of the input word as follows:

(1) If the root class (c) of the word form under consideration designates an irregular root structure (i.e., $c \neq 0$), then the computed root letters of the input word must exhibit the irregularity associated with the value of c.

(2) If the root class (c) of the word form under consideration designates a regular root structure (i.e., $c = 0$), then the root letters of the input word must not exhibit any irregular characteristics.

The root letters (1_r, 2_r, 3_r, 4_r) are computed by *variable* mapping functions (Ψ) as follows:

$$1_r = \Psi_{CC_r^*}(1) \qquad (36)$$
$$2_r = \Psi_{CC_r^*}(2) \qquad (37)$$
$$3_r = \Psi_{CC_r^*}(3) \qquad (38)$$
$$4_r = \Psi_{CC_r^*}(4) \qquad (39)$$

where the variable lists are the input word (C) and a given word form (C_r^*). Table 11 lists the root structure characteristics for each root class. To meet Condition (1) above, the root letters of the input word must meet these characteristics as follows:

(a) where the table lists an alphabetic character (including &), the corresponding root character must be the same. However, where the table lists an alphabetic character (including &) and the corresponding root letter is zero value (*), replace the * of the root with the alphabetic character from the table.

(b) Where the table lists a zero value (*), the corresponding root letter must be zero value (*).

(c) Where the table lists an arbitrary character (—), the corresponding root letter can be any alphabetic character (including & and @) but not zero value (*).

(d) Where the table lists two equal signs (=), the two corresponding characters of the root must be identical alphabetic characters (including & and @) but not zero value (*).

To meet Condition (2) above, the root letters of the input word must not meet Condition (1) for any irregular root structure listed on Table 11 when the root class (c) of the word form under consideration designates a regular root structure (i.e., $c = 0$).

To formalize this, let the sign ": :" when used between two symbols such as A : : c have the following meaning, "the root letters of A correspond to root Structure c as defined in conditions (1) and (2) above." If the computed

TABLE 11
Irregular Root Classes

(C_r) Root Class	Root Letters (I_r)			
	1	2	3	4
1	C	—	—	*
2	S	—	—	*
3	&	—	—	*
4	Z	—	—	*
5	W	—	—	*
6	Y	—	—	*
7	N	—	—	*
8	N	—	N	*
9	A	—	—	*
10	N	—	H	*
11	A	—	H	*
12	—	—	H	*
13	C	—	H	*
14	S	—	H	*
15	&	—	H	*
16	Z	—	H	*

(C_r) Root Class	Root Class (I_r)			
	1	2	3	4
17	C	=	=	*
18	S	=	=	*
19	&	=	=	*
20	Z	=	=	*
21	—	=	=	*
22	—	W	—	*
23	C	W	—	*
24	S	W	—	*
25	&	W	—	*
26	Z	W	—	*
27	—	Y	—	*
28	C	Y	—	*
29	S	Y	—	*
30	&	Y	—	*
31	Z	Y	—	*
32	—	—	—	—
33	—	—	—	*

— = arbitrary alphabetic character (including & and @), * = zero value, = = identical alphabetic characters (including & and @).

root letters of the input word, C, are compared to Root Conditions, c_r, the root structure conditions are met when

$$C :: c_r \qquad (40)$$

The value of c_r is computed from a mapping function as follows

$$c_r = \Psi_{10}(4, r) \qquad (41)$$

However, it will be noted that c_r is not uniquely defined in Table 10, but is defined only by sets of values. The operational function (Ω) tests each member of the set. Expression (40), therefore, must be modified as follows

$$C :: c_{r_i} \qquad (42)$$

where

$$c_{r_i} = \Psi_{15}(c_r, i) \qquad (43)$$

and c_r is defined in expression (41).

Expression (42) is the third restraint on expression (7).

(4) *Lexical Conditions*

The Operational Function (Ω) considers only those parsings of a word that are in the dictionary. A given parsing of an input word defines the root by the orthographic values of 1_r, 2_r, 3_r, and 4_r; and it defines the stem by the values of a_r, and b_r. This requirement is met, if an entry is contained in Hebrew Dictionary, Δ, under Root

$1_r, 2_r, 3_r, 4_r$, for Inflection Class a_r, Stem Class b_r, Root Class c_r, and "Logeme Class" d_r.

The above condition is met when

$$S_{a_r b_r c_r d_r 1_r 2_r 3_r 4_r} \in \Delta \tag{44}$$

where Δ is the class of all existing Hebrew stems (i.e., the dictionary). The dictionary, Δ, is not listed herein, but any standard Hebrew dictionary can be used for manual operation of the algorithm. A sample dictionary is contained in the Appendix. Special dictionary entries are required for certain irregular words.

Expression (44) is the fourth restraint on expression (7). The values of a_r and b_r are computed from fixed mapping functions as follows

$$a_r = \Psi_{10}(2, r) \tag{45}$$
$$b_r = \Psi_{10}(3, r) \tag{46}$$

The values of 1_r, 2_r, 3_r, and 4_r are computed from variable mapping functions previously defined in expressions (36) through (39).

C. *The Complete Operational Function* (Ω)

The computations of operational function (Ω) select from universe U, as defined by expression (17), that set of word forms that simultaneously meet the above four conditions. This is expressed as

$$\Omega(C) = \left\{ C_i^* \left| \begin{array}{l} C : C_i^* \\ a_r \in \{a_m\} \wedge \{a_n\} \wedge \{a_q\} \wedge \{a_t\} \wedge \{a_w\} \\ d_t \in \{d_m\} \wedge \{d_n\} \wedge \{d_q\} \wedge \{d_r\} \wedge \{d_w\} \\ e_w \in \{e_n\} \wedge \{e_t\} \\ C : : c_{r_t} \\ S_{a_r b_r c_r d_r 1_r 2_r 3_r 4_r} \in \{\Delta\} \end{array} \right. \right\}$$
$$= \{C_\omega^*\} \tag{47}$$

where Δ is the dictionary, and the values of S, a, b, c, d, e, 1, 2, 3, and 4 are computed by the mapping functions (Ψ). In this expression, $\{C_\omega^*\}$, is the list of all valid sets of constituent members of the input word as shown in its expanded form in expression (5).

D. *The Output of the Algorithm*

The output of the algorithm is a set of values for the grammatical attributes of each computed set of constituent members of the input word, i.e., for each output of the operation function (Ω). For each such set of computed constituent members the operation function produces an output (C_ω^*) which, in its expanded form is, from expressions (7) and (47)

$$C_\omega^* = D_{k_\omega} Z_{l_\omega} Y_{m_\omega} B_{q_\omega} S_{r_\omega}{}^* A_{t_\omega} P_{w_\omega} E_{x_\omega} \tag{48}$$

The output of the algorithm for each C_ω^* is computed as follows:

$$a_\omega = \Psi_{10}(2, r) = a_{r_\omega} \tag{49}$$
$$b_\omega = \Psi_{10}(3, r) = b_{r_\omega} \tag{50}$$
$$c_\omega = \Psi_{10}(4, r) = c_{r_\omega} \tag{51}$$
$$d_\omega = \Psi_8(3, t) = d_{t_\omega} \tag{52}$$
$$e_\omega = \Psi_9(4, w) = e_{w_\omega} \tag{53}$$
$$g_\omega = \Psi_3(4, n) = g_{h_\omega} \tag{54}$$
$$h_\omega = \Psi_4(4, m) = h_{m_\omega} \tag{55}$$
$$j_\omega = \Psi_5(2, l) = j_{l_\omega} \tag{56}$$
$$u_\omega = \Psi_1(2, k) = u_{k_\omega} \tag{57}$$
$$v_\omega = \Psi_2(2, w) = v_{wx_\omega} \tag{58}$$
$$1_\omega = \Psi_{CC^*}(1) = 1_{r_\omega} \tag{59}$$
$$2_\omega = \Psi_{CC^*}(2) = 2_{r_\omega} \tag{60}$$
$$3_\omega = \Psi_{CC^*}(3) = 3_{r_\omega} \tag{61}$$
$$4_\omega = \Psi_{CC^*}(4) = 4_{r_\omega} \tag{62}$$

It will be observed that all these values have been computed by and are available from the operational function (Ω). Thus no new computation of them is required. This completes the formal description of the algorithm. The next section describes a computer program of the algorithm.

IV. COMPUTER PROGRAM FOR ALGORITHM

This algorithm reduces the task of analyzing Hebrew words to a sequence of table look-up operations. However, it is much more complex than the algorithm for generating Hebrew words. This section describes a computer program of the algorithm for analyzing Hebrew words. The program is presented in flow-chart form from which it can be coded for any given machine. However, the flow chart omits obvious mechanical details in order to make it easier for manual operation. The over-all view of the program is discussed first, following which, a detailed description is given of each section.

A. *An Over-all View of the Program*

Figure 2 is an over-all view of the program. The operations of each block are related to the formal statement of the algorithm in expression (47) and are given in detail in the designated figures.

Block 1 divides the input text into words, and feeds one

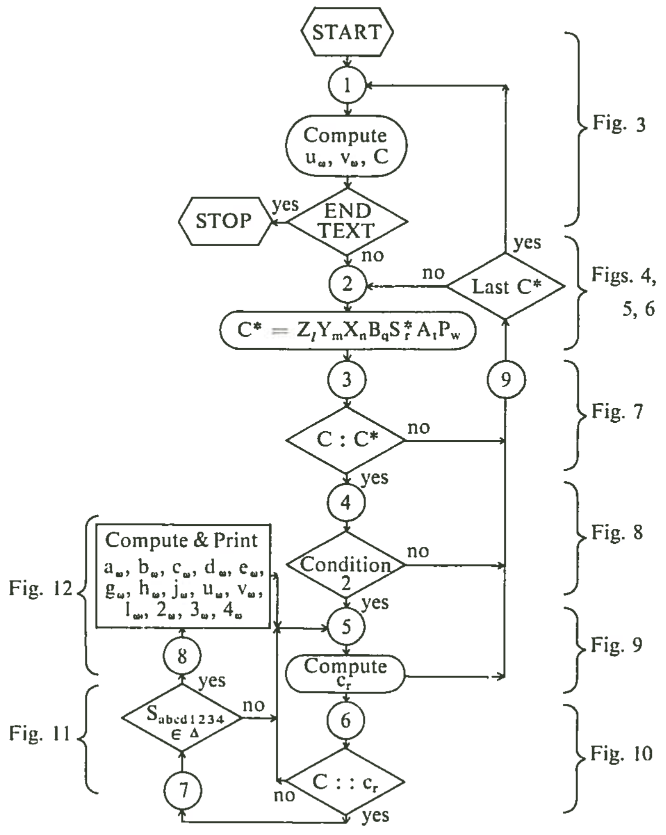

START
1
Compute u_ω, v_ω, C
STOP
yes
END TEXT
no
2
no
Last C*
yes
$C^* = Z_l Y_m X_n B_q S^*_r A_t P_w$
3
9
C : C*
no
yes
4
Compute & Print $a_\omega, b_\omega, c_\omega, d_\omega, e_\omega, g_\omega, h_\omega, j_\omega, u_\omega, v_\omega, 1_\omega, 2_\omega, 3_\omega, 4_\omega$
Condition 2
no
yes
5
8
yes
Compute c_r
$S_{abcd1234} \in \Delta$
no
6
7
C : : c_r
no
yes
Fig. 3
Figs. 4, 5, 6
Fig. 7
Fig. 8
Fig. 9
Fig. 10
Fig. 12
Fig. 11

Fig. 2. Flow Chart of Word Analysis Algorithm.

word (C) at a time for analysis. Block 2 generates word forms, C^*_i, which are presented one at a time to be tested against the six conditions imposed on expression (47). Word forms that fail to meet the conditions of Blocks 3, 4, 6, and 7, are immediately rejected and a new word form is generated by Block 2.

Block 3 tests the given word form against the input word, C, for the first condition as given in expression (18), namely,

$$C : C^*_i$$

Block 4 tests those word forms that pass the first condition against the next three conditions as given in expression (35), namely

$$a_r \in \{a_m\} \wedge \{a_n\} \wedge \{a_q\} \wedge \{a_t\} \wedge \{a_w\}$$
$$d_t \in \{d_m\} \wedge \{d_n\} \wedge \{d_q\} \wedge \{d_r\} \wedge \{d_w\}$$
$$e_w \in \{e_r\} \wedge \{e_t\}$$

For those word forms that meet these conditions, Block 5 computes a value for root class, c_r, which it presents to Block 6 for test of the fifth condition as given in expression (42), namely,

$$C : : C_{r_i}$$

For those word forms that meet the first three conditions, Block 7 computes the value of attributes a, b, c, d, 1, 2, 3, and 4, and checks these values against the fourth condition as given in expression (44), namely,

$$S_{abcd1234} \in \Delta$$

For word forms that meet all six conditions, Block 8 computes the value of all grammatical attributes and prints them as one analysis of the input word, after which a new word form is generated by Block 2 and the procedure is repeated until all possible parsings of the input word are printed. When this is completed, a new input word is presented by Block 1 until all input words are parsed.

The next sections describe the operations of each block in detail.

B. *Block 1: Word Division*

Block 1 of Figure 2 divides the input text into words and assigns a sequence number to each word. Each word is presented one at a time for analysis.

Figure 3 is a detailed flow chart of Block 1 of Figure 2. In it the following operations are performed:

(1) The input text is read one character at a time.

(2) Each character is examined to determine whether it is: (a) left punctuation, (b) space, (c) right punctuation, or (d) alphabetic character.

(3) If the character is a left punctuation [i.e., character $= \Psi 1$ (1, k)], a value is computed for u_ω and the next character is examined.

(4) If the character is a space, a check is made to see if there was a word immediately preceding; if so ($N_c \neq 0$), the end of a word is indicated; if not ($N_c = 0$), the space is superfluous, and the next character is examined.

(5) If the character is a right punctuation mark [character $= \Psi 2$ (1, x)], the value of v_ω is computed and a check is made to see if the mark was immediately preceded by a word; if so ($N_c \neq 0$) the end of that word is sensed and control passes to Block 2, of Figure 2, if not ($N_c = 0$), it is assumed that another right punctuation preceded this one and the identity of the present punctuation mark, v_ω, is computed and printed as a separate word.

(6) If the character is an alphabetic, the character is stored in array C, and the character counter (N_c) is indexed one count.

(7) When the end of a word is sensed, operations in Block 1 cease until all parsings of that word are completed.

When the end of a word is sensed its characters, less left and right punctuation, if any, are contained in sequence in array C, and the number of its characters is contained in the input word character counter N_c. The punctuation marks are unique characters; they are sensed and removed in the first operation. This reduces the

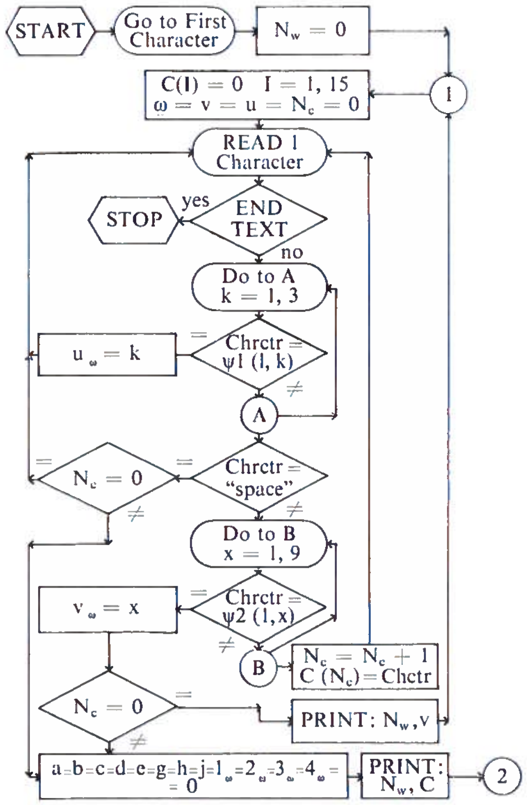


Fig. 3. Flow Chart of Input Word Divider.

number of word forms to be generated in Block 2 by a factor of 2 × 8 = 16, thus reducing the over-all computation time.

C. *Block 2: Word Form Generator*

Figure 4, 5, and 6 comprise a detailed flow chart of the word-form generator, Block 2 of Figure 2. This operation constructs every word form that might meet the four conditions imposed on expression (47). Several techniques are used to limit the forms generated to a relatively small number. Otherwise the alternative would be to generate all possible combinations of the indices *l*, m, n, q, r, t, u, v, and w, which would produce over 22 million word forms, an unrealistic undertaking.

The following economizing techniques are used:

(1) Only word forms that have the same number of characters as the input word are constructed. This is accomplished by computing the required number of characters in the stem form S* and by using only that section of Table 10 containing the computed number of characters.

(2) Invalid combinations of word constituents are not constructed. This is accomplished by organizing the look-up tables in such a way that tables with high hierarchical rank specify valid sections of tables with lower hierarchical rank so that only valid sections are used. It was possible to organize the tables so that two of the three restraints of expression (35) and part of the third could be imposed immediately, namely

$$\begin{aligned} &a_r = a_m = a_n = a_q = a_t = a_w \\ &d_t \in \{d_m\} \Lambda \{d_n\} \Lambda \{d_q\} \\ &e_w \in \{e_r\} \Lambda \{e_t\} \end{aligned} \qquad (63)$$

Thus only combinations of word constituents that meet the above restraints are generated.

(3) Newly selected word constituents are immediately compared to the corresponding characters of the input word. Only those that match the input word are considered for combination with lower level constituents. Rejected high level constituents prevent the construction of further combinations of lower level constituents under its control. Thus unfruitful paths are terminated and only fruitful paths are pursued.

The hierarchy of the word constituents are arranged as follows:

Level I: P_w, Z_l
Level II: A_t
Level III: Y_m, X_n, B_q, S^*_r

Constituent Z_l is independent of the other word constituents and does not enter into the restraints on the algorithm. Constituent P_w governs A_t with respect to grammatical attributes a and e. Each P_w of Table 9 specifies a t_{min} and t_{max} for A_t of Table 8, so that

$$a_t = a_w \quad \text{and} \quad e_w \in \{e_t\} \qquad (64)$$

Constituent A_t governs constituents Y_m, X_n, and B_q with respect to grammatical attributes a and d. Each A_t of Table 8 specifies m_{min}, m_{max}, n_{min}, n_{max}, q_{min}, and q_{max} for Tables 5, 6, and 7 respectively, so that

$$a_t = a_m = a_n = a_q \quad \text{and} \quad d_t \in \{d_m\} \Lambda \{d_n\} \Lambda \{d_q\} \qquad (65)$$

Constituent A_t also governs the stem form S^*_r with respect to grammatical attributes a and e so that r_{min} and r_{max} are computed for Table 10 such that

$$a_t = a_r$$

and (66)

$$e_w \in \{e_t\} \wedge \{e_r\}$$

Combining expression (64), (65), and (66) produce the results of expression (63).

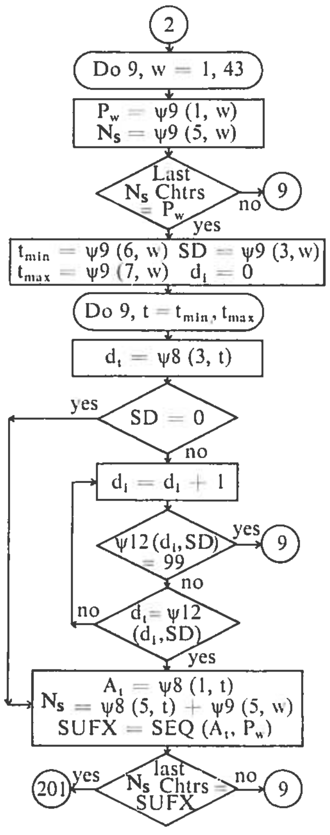


Fig. 4. Flow Chart of Suffix Generator.

In Figure 4, the following operations are performed:

(1) Each value of P_w in Table 9 is compared with the last characters of the input word C. For values of P_w that do not match the last characters of C, no further combinations are made.

(2) For each value of P_w that matches the last characters of C, a value of t_{min}, t_{max}, SD, and the number of characters in P_w is computed from Table 9. The values of t_{min} and t_{max} specify that section of Table 8 containing values of A_t that meet the requirement

$$a_t = a_w$$

and

$$e_w \in \{e_t\}$$

for the given P_w. The value of SD specifies a set of values of d that are valid for the given P_w.

(3) For each value of t between t_{min} and t_{max} the value of d_t is computed from Table 8 and compared to the list of values in $\{d_w\}$ as listed in horizontal row SD of Table 14. This tests for the condition

$$d_t \in \{d_w\}$$

The test is bypassed if SD = 0, since all values of d are in $\{d_w\}$ for this case. If the value of d_t is not in $\{d_w\}$, all further computations cease for the given combination A_tP_w.

(4) For those values of t where

$$d_t \in \{d_w\}$$

the value of A_t is computed from Table 8 and a suffix is constructed from the given P_w and A_t using the function SEQ which places the characters of its arguments in the specified array in sequence with no intervening spaces. Also the number of characters (N_s) in the suffix is computed.

(5) The suffix constructed in (4) is compared to the last characters of the input word C. For suffixes that do not match, no further computations are made for the given combination A_tP_w.

(6) In Figure 5, for each suffix that matches the last characters of C, values of q_{min}, q_{max}, n_{min}, n_{max}, m_{min}, m_{max}, and a are computed from Table 8. The values of q_{min}, etc., specify the sections of Tables 4, 5, and 7 containing Y_m, X_n, and B_q that meet the requirements

$$a_t = a_m = a_n = a_q$$

and

$$d_t \in \{d_m\} \wedge \{d_n\} \wedge \{d_q\}$$

for the given value of t.

In Figures 5 and 6 the following operations are performed:

(1) For each suffix that matches the last characters of the input word C, each value of Z_l from Table 5 is compared to the first character of the input word C. For the values of Z_l that do not match, no further computations are made.

(2) For each value that does match, a prefix is constructed from the given Z_l and each Y_m in Table 4 from m_{min} to m_{max} using function SEQ, and the number of characters (N_p) in the prefix is computed.

(3) Each prefix constructed in (2) is compared to the first characters of the input word C. For those prefixes that do not match the first characters of C, no further computations are made for the respective Z_lY_m combination.

(4) For each prefix that does match the first characters

of the input word, new prefixes are constructed from the given $Z_l Y_m$ combination and each value of X_n from Table 3 from n_{min} to n_{max}, and the number of characters in the new prefix is computed.

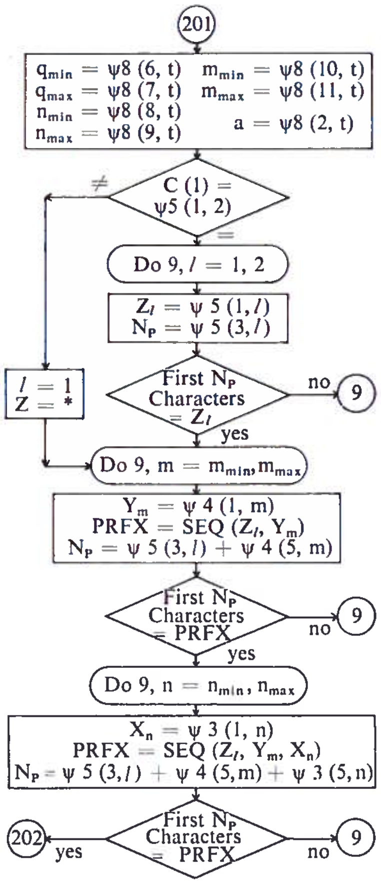

201
$q_{min} = \psi 8\ (6, t)$
$q_{max} = \psi 8\ (7, t)$
$n_{min} = \psi 8\ (8, t)$
$n_{max} = \psi 8\ (9, t)$
$m_{min} = \psi 8\ (10, t)$
$m_{max} = \psi 8\ (11, t)$
$a = \psi 8\ (2, t)$
$C (1) = \psi 5 (1, 2)$
≠
=
Do 9, l = 1, 2
$Z_l = \psi\ 5\ (1, l)$
$N_P = \psi\ 5\ (3, l)$
$l = 1$
$Z = *$
First N_P Characters $= Z_l$
no
yes
9
Do 9, $m = m_{min}, m_{max}$
$Y_m = \psi\ 4\ (1, m)$
$PRFX = SEQ\ (Z_l, Y_m)$
$N_P = \psi\ 5\ (3, l) + \psi\ 4\ (5, m)$
First N_P Characters = PRFX
no
yes
9
Do 9, $n = n_{min}, n_{max}$
$X_n = \psi\ 3\ (1, n)$
$PRFX = SEQ\ (Z_l, Y_m, X_n)$
$N_P = \psi\ 5\ (3, l) + \psi\ 4\ (5, m) + \psi\ 3\ (5, n)$
First N_P Characters = PRFX
202
yes
no
9

Fig. 5. Flow Cart of Prefix Generator.

(5) Each prefix constructed in (4) is compared to the first characters of the input word C. For those prefixes that do not match, no further computations are made for the respective $Z_l Y_m X_n$ combination.

(6) For each prefix that does match the first characters of the input word, new prefixes are constructed from the given $Z_l Y_m X_n$ combination and each value of B_q from Table 7 from q_{min} to q_{max}, and the number of characters in the new prefix is computed.

(7) Each prefix constructed in (6) is compared to the first characters of the input word C. For those prefixes that do not match, no further computations are made for the respective $Z_l Y_m X_n B_q$ combination. Computations that reach this point produce a combination of word constituents P_w, A_t, Z_l, Y_m, X_n, and B_q that match the corresponding part of the input word and meet the following conditions

$$a_w = a_t = a_q = a_m = a_n$$
$$d_t \in \{d_q\} \wedge \{d_m\} \wedge \{d_n\} \wedge \{d_w\}$$
$$e_w \in \{e_t\}$$

For each such combination, computations continue in Figure 6.

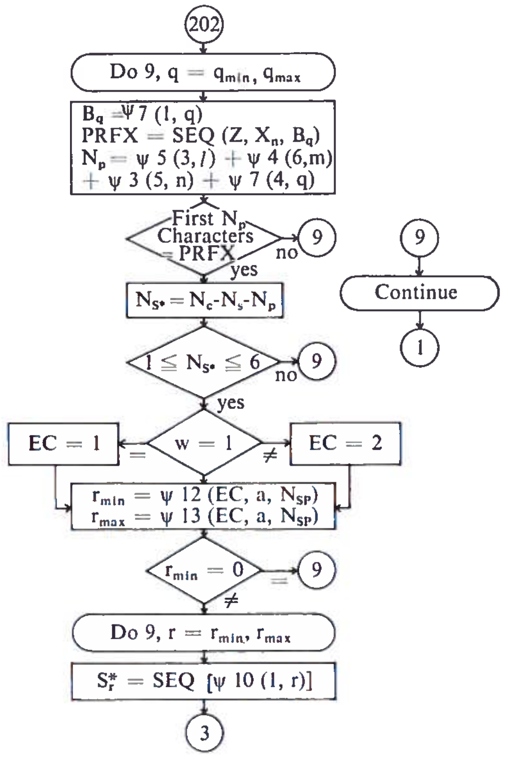

202
Do 9, $q = q_{min}, q_{max}$
$B_q = \Psi 7\ (1, q)$
$PRFX = SEQ\ (Z, X_n, B_q)$
$N_p = \psi\ 5\ (3, l) + \psi\ 4\ (6, m) + \psi\ 3\ (5, n) + \psi\ 7\ (4, q)$
First N_p Characters = PRFX
no
yes
9
$N_{S*} = N_c - N_s - N_p$
$1 \leq N_{S*} \leq 6$
no
yes
9
$w = 1$
=
≠
EC = 1
EC = 2
$r_{min} = \psi\ 12\ (EC, a, N_{SP})$
$r_{max} = \psi\ 13\ (EC, a, N_{SP})$
$r_{min} = 0$
=
≠
9
Do 9, $r = r_{min}, r_{max}$
$S_r^* = SEQ\ [\psi\ 10\ (1, r)]$
3
9
Continue
1

Fig. 6. Flow Chart of Stem Generator.

In Figure 6 the following additional operations are performed:

(1) For each $P_w A_t Z_l Y_w X_n B_q$ combination that meet the above requirements, the number of characters (N_{S*}) required in stem form S* is computed as the number of characters (N_c) in the input word C less the number of characters (N_s) in the given suffix $A_t P_w$, less the number of characters (N_p) in the given prefix $Z_l Y_m X_n B_q$.

(2) The magnitude of the number of characters in the stem form S* is tested against the requirement

$$1 \leqq N_{S*} \leqq 6$$

If the value of N_{S*} exceeds these limits, no further computations are made for the given $P_w A_t Z_l Y_n X_m B_q$ combination since stem forms contain from one to six characters only.

(3) If N_{S*} is of the correct magnitude, a value is computed for EC, r_{min} and r_{max} from Tables 12 and 13 respectively. The value of N_{S*} specifies the vertical column in Tables 12 and 13 that contain values of r_{min} and r_{max}

TABLE 12

Value of r_{min}

a	EC = 1						EC = 2					
	$N_{S_r^*}$						$N_{S_r^*}$					
	(1)	(2)	(3)	(4)	(5)	(6)	(1)	(2)	(3)	(4)	(5)	(6)
1	1	5	73	224	409	547	1	11	84	229	409	547
2	0	45	173	313	461	560	3	53	195	348	490	588
3	4	60	218	382	521	0	4	67	220	396	533	0

TABLE 13

Value of r_{max}

a	EC = 1						EC = 2					
	$N_{S_r^*}$						$N_{S_r^*}$					
	(1)	(2)	(3)	(4)	(5)	(6)	(1)	(2)	(3)	(4)	(5)	(6)
1	1	36	161	310	460	559	2	44	172	312	460	559
2	0	59	213	381	520	611	3	59	217	381	520	611
3	4	72	223	408	544	0	4	72	223	408	546	0

such that all stem forms in Table 10 between those limits contain N_{S*} characters. The value of EC specifies the section of the table and the value of *a* specifies the horizontal row of Tables 12 and 13 that contain values of r_{min} and r_{max} such that

$$a_r = a_t$$
$$\{e_r\} = \{e_t\}$$

Thus the computed values of r_{min} and r_{max} specify a section of Table 10 that meets all the following requirements.

(a) the number of characters in S_r^* is equal to the number of characters in S

(b) $a_r = a_t = a_m = a_n = a_q = a_w$

(c) $e_w \in \{e_t\} \land \{e_r\}$

(4) Each S_r^* in Table 10 between r_{min} and r_{max} from (3) above is selected one at a time for further computation in Block 3 of Figure 2.

Since word constituents P, A, Z, Y, X, and B may be zero value, all possible valid combinations of these constituents that correspond to the input word are generated by the above procedure.

D. *Block 3: Test of Word-Form Conditions*

Block 3 of Figure 2 tests the given word-form against the condition

$$C : C^*$$

Part of this condition was determined in Block 2 of Figure 2 in order to limit the number of word-form combination; there it was determined that the number of characters of C* is the same as C and that all characters of C* except those of constituent S_r^* are identical to the corresponding characters of C. Thus the only computation required by Block 3 to complete the test of this condition is

$$S : S_r^*$$

where S is the characters of C not corresponding to the given prefix and suffix. Figure 7 is a detailed flow chart of the operations of Block 3 of Figure 2.

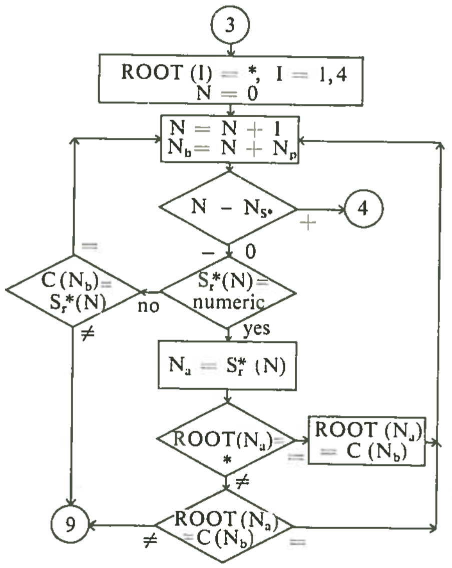


Fig. 7. Flow Chart for Test of Word-Form Conditions.

In Figure 7 the following operations are performed for each S_r^* produced in Figure 6:

(1) The four characters of array ROOT are set to an initial value of zero (*).

(2) The Nth character of S_r^* is examined.

(3) If the Nth character is a numeral, say N_a, the character in ROOT (N_a) is tested to see that it does not already contain an alphabetic character as may be the case for stem forms such as 1W33. If it does not [i.e., ROOT (N_a) = *], the (N + N_p) th character of the input word C is placed in ROOT (N_a). If it does contain an alphabetic character, the (N + N_p)th character of the input word C is compared to the character; if they are identical testing continues, if not, computations cease for the given combination $Z_l Y_m X_n B_q S_r^* A_t P_w$.

(4) If the Nth character of S_r^* is not a numeral, it is compared to the (N + N_p) th character of the input word C. If the characters are not identical, all further computations cease for the given combination $Z_l Y_m X_n B_q S_r^*$-$A_t P_w$.

(5) If the characters are identical, index N is increment-

TABLE 14

Values of d

SD	I_d																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
01	10	12	13	19	99															
02	11	15	16	17	18	20	21	22	23	24	25	26	27	28	29	32	33	37	38	99
03	20	21	22	23	24	25	26	27	28	29	99									
04	32	33	37	38	99															
05	10	11	12	13	14	15	16	17	18	19	99									
06	17	18	99																	
07	20	21	22	23	25	26	27	28	29	99										
08	23	25	27	99																
09	26	28	99																	
10	20	21	22	29	99															
11	15	16	33	37	99															
12	12	13	14	18	19	99														
13	10	32	99																	
14	10	11	15	16	99															
15	32	33	37	99																
16	10	11	12	13	14	15	16	17	18	19	23	25	27	33	37	99				
17	20	21	22	24	26	28	29	32	38	99										
18	10	11	12	13	14	15	16	17	18	19	33	37	99							
19	32	38	99																	
20	20	21	22	24	26	28	29	99												
21	20	21	22	23	24	25	26	27	28	29	32	33	37	38	99					
22	10	11	12	13	14	15	16	17	18	19	26	28	99							
23	20	21	22	23	25	27	29	32	33	37	38	99								
24	15	16	23	25	27	33	37	99												
25	12	13	14	17	18	19	26	28	99											
26	10	20	21	22	24	29	32	99												
27	20	21	22	24	29	99														
28	15	16	99																	
29	33	37	99																	
30	20	21	22	23	25	27	29	99												
31	26	28	38	99																
32	23	25	27	32	33	37	99													
33	15	16	23	25	27	33	37	99												
34	12	13	14	17	18	19	26	28	38	99										
35	10	11	12	13	14	15	16	17	18	19	32	33	37	38	99					
36	12	13	14	17	18	19	38	99												
37	10	11	15	16	32	33	37	38	99											
38	10	11	15	16	32	33	37	99												
39	12	13	17	18	99															
40	21	99																		
41	10	12	99																	
42	13	17	18	99																
43	12	13	17	18	21	99														
44	16	99																		
45	10	11	12	13	14	15	17	99												
46	13	17	18	99																
47	12	13	17	99																
48	10	11	99																	
49	12	13	99																	
50	17	18	99																	
51	10	11	99																	
52	15	16	99																	
53	11	12	13	14	15	16	17	18	19	23	25	26	27	28	29	99				
54	10	11	14	15	16	19	20	21	24	25	26	29	99							
55	10	11	20	21	24	29	99													
56	14	15	16	19	25	26	99													
57	10	11	12	14	99															
58	20	21	22	32	99															
59	20	21	22	24	29	32	99													
60	10	11	12	13	14	15	16	17	18	19	23	25	26	27	28	33	37	38	99	
61	12	13	17	18	21	99														
62	12	13	18	21	99															
63	12	13	17	21	99															
64	12	13	21	99																
65	17	99																		
66	18	99																		
67	10	11	12	13	15	16	17	18	23	25	26	27	28	33	37	38	99			
68	10	99																		
69	11	99																		
70	12	99																		
71	20	99																		
72	21	99																		
73	24	99																		
74	32	99																		
75	38	99																		
76	10	11	14	15	16	99														
77	11	15	99																	
78	14	15	16	99																
79	16	18	99																	

ed by 1 and steps (2) and following are repeated until computations terminate or index N exceeds N_{s*}.

(6) When index N exceeds N_{s*}, all characters of S^*_r have passed the conditions

$$S : S^*_r$$

and computations continue into Block 4 of Figure 2.

(7) In addition, array ROOT now contains the root letters of C as specified by the given C^*_r. These values are required in a later test, but are computed most easily at this point.

E. *Block 4: Test of Grammatical Conditions*

Block 4 of Figure 2 tests the given word-form against the grammatical conditions of expression (35). However most of these conditions are imposed by Block 2 in order to limit the number of word-form combinations. These previously imposed conditions are given in expression (67). The only additional conditions required to satisfy expression (35) is

$$d_t \in \{d_r\}$$

This final grammatical test is made in Block 4.

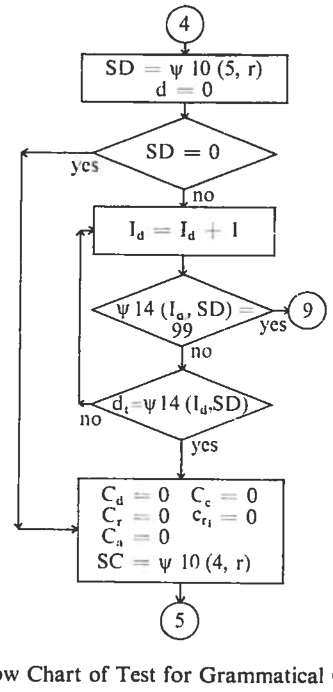


Fig. 8. Flow Chart of Test for Grammatical Conditions.

Figure 8 is a detailed flow chart of the operations of Block 4 of Figure 2. The following computations are made:

(1) For the given value of r from Figure 7, the value of SD is computed from Table 10. This value specifies a set of values of d, $\{d_r\}$, that are valid for the given stem-form S^*_r, which set is contained in horizontal row SD of Table 14.

(2) The given value of d_t as computed in Figure 4 is compared to each element of $\{d_r\}$. If d_t is not an element of $\{d_r\}$ computations cease for the given combination $Z_l Y_m X_n B_q S^*_r A_t P_w$.

This test is by-passed if SD = 0 since $\{d_r\}$ contains all values of d in this case.

(3) If $d_t \in \{d_r\}$, the value of SC is computed from Table 10 which defines the set of values of c that are valid for the given stem-form S^*_r. This set $\{c_r\}$ is listed on horizontal line SC of Table 15. Certain initial conditions are set and computation advances to Block 5 of Figure 2.

F. *Block 5: Computation of c_r*

Block 5 of Figure 2 computes values of c_r from Table 15 that are valid for the given stem-form S^*_r. These values are presented one at a time to Block 6 for testing the root condition

$$C :: c_r$$

The following conditions determine how the value of c_r is computed.

(1) If SC is not zero, its value specifies the set of values of c that are valid for the given stem-form S^*_r. In this case only the values in the specified set are computed. These values are arranged in the table in such order that the last value meeting the requirement

$$C :: c_r$$

is the correct value.

(2) If SC is zero, all possible values of c must be tested. However, in order to avoid missing certain alternate parsings, these values are divided into two sets contained in horizontal rows 15 and 44 of Table 15 respectively. A separate run is made on each set. The values in each set are arranged such that the last value meeting the requirements

$$C :: c_r$$

is the correct value. If none of the values in either set meets the requirements, the regular root class (c = 33) is tested.

(3) The variable c_c is used to monitor the progress of computation. If $c_c = 0$, computation is in the initial phase. If $c_c = 1$, row 15 is completed and row 44 is in progress. If $c_c = 2$, both rows are completed.

(4) All nouns (a = 3) meet the requirement $C :: c_r$ and bypass Blocks 5 and 6 of Figure 2.

TABLE 15
Values of c

SC	c_a																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
01	09	11	99																
02	22	23	24	25	26	99													
03	21	17	18	19	20	99													
04	27	28	29	30	31	99													
05	12	13	14	15	16	99													
06	12	13	14	15	16	11	99												
07	12	13	14	15	16	10	11	99											
08	01	02	03	04	99														
09	17	18	19	20	99														
10	01	02	99																
11	12	10	11	99															
12	23	24	99																
13	28	29	99																
14	13	14	99																
15	01	02	03	04	05	06	07	08	09	12	13	14	15	16	10	11	21	17	18
												19	20	22	23	24	25	26	99
16	05	06	07	99															
17	21	17	18	19	20	22	23	24	25	26	99								
18	21	22	27	99															
19	17	18	23	24	28	29	99												
20	19	25	30	99															
21	20	26	31	99															
22	07	08	99																
23	05	06	07	08	99														
24	03	99																	
25	04	99																	
26	07	99																	
27	08	99																	
28	09	99																	
29	10	99																	
30	11	99																	
31	15	99																	
32	16	99																	
33	17	99																	
34	18	99																	
35	19	99																	
36	20	99																	
37	22	99																	
38	25	99																	
39	26	99																	
40	27	99																	
41	30	99																	
42	31	99																	
43	32	99																	
44	27	28	29	30	31	32	99												

(5) The variable c_{r_1} is used to retain the most recent value of c_r meeting the condition

$$C :: c_r$$

(6) The variable c_d is used to indicate whether any value of c in row 15 of Table 15 met the above requirement. It $c_d = 0$, no value of c met the requirement; if $c_d = 1$, some value did meet the requirement.

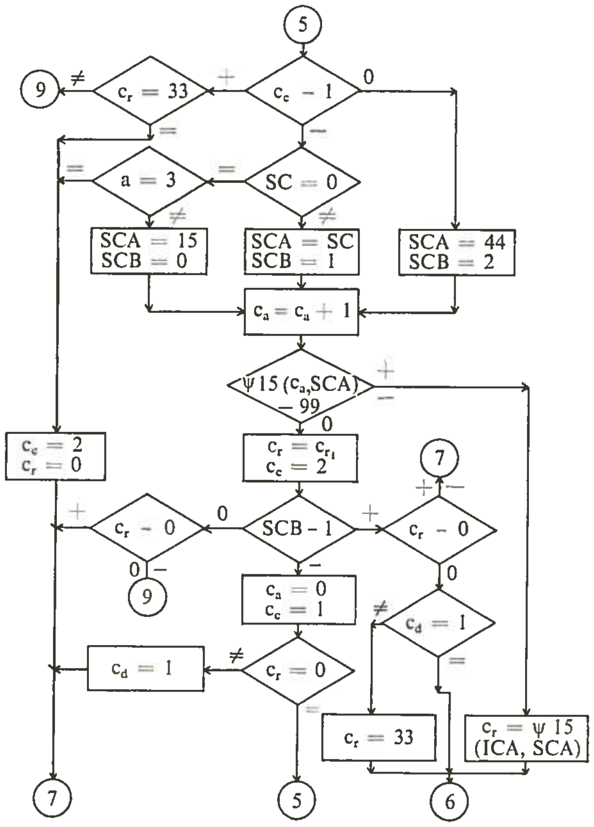


Fig. 9. Flow Chart for Computation of c_r.

Figure 9 is a detailed flow-chart of Block 5 of Figure 2. In this figure, the following computations are made:

(1) The value of c_c is examined. If $c_c = 0$, the value of SC is examined. If SC $\neq 0$, c_r is set to each value in horizontal row SC of Table 15 one at a time and presented to Block 6 for test of the condition

$$C :: c_r$$

When all values in row SC have been tested [i.e., $\Psi 15$ $(C_a, SC) = 99]$, c_r is set to the value of c_{r_1}, c_c is set to 2, and the value of c_r is examined. It $c_r \neq 0$ some value in row SC met the requirement, so computation is advanced to Block 7. If $c_r = 0$, no value in row SC met the requirement and computation ceases for the given combination $Z_l Y_m X_n B_q S^r_* A_t P_w$.

(2) If $c_c = 0$ and SC $= 0$, the value of a is examined. If $a = 3$ (i.e., the input word is considered a noun), further computation in Blocks 5 and 6 is by-passed, c_c is set to 2, c_r is set to zero, and computation advances to Block 7.

(3) If $c_c = 0$, SC $= 0$ and $a \neq 3$, c_t is set to each value in horizontal row 15 of Table 15 one at a time and presented to Block 6 for test of the condition

$$C :: c_r$$

When all values in row 15 have been tested [i.e., $\Psi 15$ $(c_a, 15) = 99]$, index c_a is reset to zero, c_c is set to the value of c_{r_1}, c_c is set to 1 and the value of c_r is examined. If $c_r \neq 0$, its value represents the last value of c_r to meet the condition

$$C :: c_r$$

In this case, c_d is set to 1, and computation advances to Block 7. If $c_r = 0$, no value of c_{r_1} met the requirement, and computation returns to the start of Block 5.

(4) If $c_c = 1$, c_r is set to each value in horizontal row 44 of Table 15 one at a time and presented to Block 6 of Figure 2 for the test of the condition

$$C :: c_r$$

When all values in row 44 have been tested [i.e., $\Psi 15$ $(c_a, 44) = 99]$, c_r is set to the value of c_{r_1}, c_r is set to 2, and the value of c_r is examined. If $c_r \neq 0$, its value represents the last value of c_r to meet the condition

$$C :: c_r$$

and computation is advanced to Block 7. If $c_r = 0$, no value of c_r in row 44 meet the requirement, in which case the value of c_d is examined. If $c_d = 0$, no value in row 15 met the requirement either, so c_r is set to 33 ($c_r = 33$ for regular root class) and is presented to Block 6 for test of the condition

$$C :: c_r$$

If $c_d = 1$, some value in row 15 did meet the requirement, so the value $c_r = 0$ is presented to Block 6.

(5) If $c_c = 2$, all values of c_r that are valid for the stem-form S^*_r have been computed and tested. In this case, the value of c_r is compared to 33. If $c_r \neq 33$, the given combination meets the root conditions for the regular root class so c_r is set to 0, and computation advances to Block 7 of Figure 2.

This completes the description of Block 5 of Figure 2.

G. *Block 6: Test of Root Conditions*

Block 6 of Figure 2 tests the given stem-form S^*_r against the root conditions

$$C :: c_r$$

for the value of c_r computed in Block 5. Figure 10 is a detailed flow chart of Block 6 of Figure 2. The variable I_r is used as an index to indicate the root letter under test. The variable I_Q is used to indicate the state of the test for equality of root characters. If $I_Q = 0$, no characters of the root have been given the test of equality. If $I_Q = 1$, one character is under test. Variable EQ is used to store the value of a root letter under test for equality.

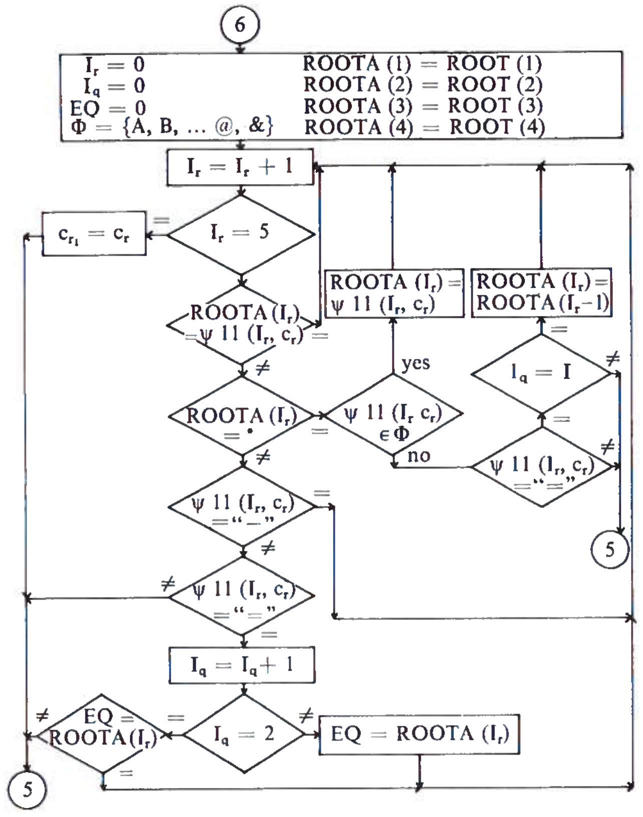


Fig. 10. Flow Chart of Test of Root Conditions.

In Figure 10, the following operations are performed:

(1) Certain indices are set to initial values and then the root letters of C which were computed in Figure 7 are examined one at a time and compared to the requirements listed on horizontal row c_r of Table 11.

(2) If the root letter under test [ROOTA (I_r)] is identical to the corresponding letter of Table 11 [Ψ11 (I_r, c_r)], the root letter passes the test and the next root letter is examined for this same test.

(3) If the root letter under test [ROOTA (I_r)] is not identical to the corresponding letter of Table 11 [Ψ11 (I_r, c_r)], it is compared with "*". If the root letter under test is "*", the corresponding character in Table 11 [i.e., Ψ11 (I_r, c_r)] is examined to determine that it is a member of the transliterated Hebrew alphabet, Φ. If it is, the root letter under test is changed to this value, it is assumed to pass the test and the next root letter is examined by test (2) above.

(4) If Ψ11 (I_r, c_r) is not a member of Φ, its value is compared to "=" to determine whether the character under test is expected to be identical to another root character. If Ψ11 (I_r, c_r) is not "=", the given c_r does not meet the condition

$$C :: c_r$$

and control returns to Block 5. If Ψ11 (I_r, c_r) is "=", the value of I_Q is examined to see if a previous root letter was expected to be identical. If $I_Q = 0$, no previous root letter was so considered, the given c_r does not meet the condition C : : c_r, and control returns to Block 5. If $I_Q = 1$, a previous root letter was expected to be identical, the root letter under consideration is made the same value, it is considered to meet the test and the next root letter is examined by test (2).

(5) If the root letter under test does not meet tests (2) through (4), the corresponding value in Table 11 [i.e., Ψ11 (I_r, c_r)] is compared to "—". If the value is "—", any value of the root letter is permissible and the root letter passes the test. The next root letter is selected and examined by test (2).

(6) If the root letter under test does not meet tests (2) through (5), the corresponding value in Table 11 [i.e., Ψ11 (I_r, c_r)] is compared to "=". If the value in Table 11 is not "=", the given stemform S^*_r does not meet the root requirements of the root class, c_r, and control is returned to Block 5 for a new value of c_r. If the value is "=", two root letters are expected to be identical. To determine whether they are, the following steps are taken:

(a) Index I_Q is advanced by 1 and compared to the value "2". If the value is less than 2, the first of the two root letters required to be identical is being considered, so its value is stored in register EQ to be used later.

(b) If the value of index I_Q is "2", the second of the two root letters required to be identical is being examined, and its value is compared to the value stored in register EQ. If they are identical, the test is passed, and the next character is examined by test (2). If they are not identical, the stem-form, $S^*_{r_i}$ does not meet the root requirements for the root class, c_r, and control returns to Block 5 for a new value of c_r.

(7) When all root letters have been examined and have passed tests (2) through (5) (i.e., $I_r = 5$), the value c_r meets all the requirements of C : : c_r so its value is placed in register c_{r_1} and control returns to Block 5.

This completes the description of Block 6 of Figure 2.

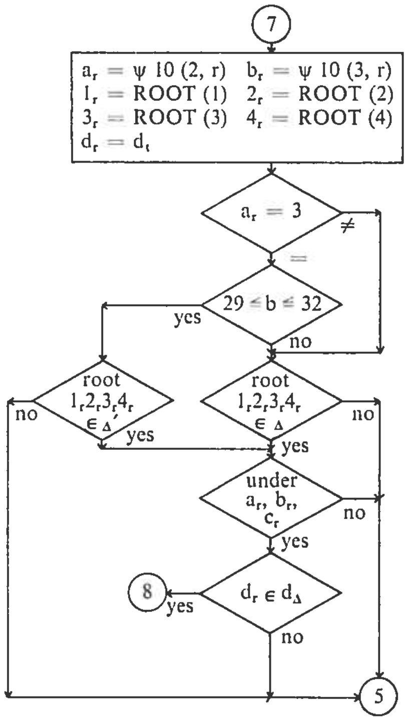
H. *Block 7: Test of Lexical Conditions*

Block 7 of Figure 2 tests the root, inflection class and stem class of the given word-form against the condition

$$S_{abcd1234} \in \Delta$$

Two dictionaries are used. The main dictionary, Δ, contains most Hebrew words; the auxiliary dictionary, Δ', contains a few irregular nouns. These irregular nouns are listed under stem forms 29 through 32.

Figure 11 is a flow chart of Block 7 of Figure 2.



7
$a_r = \psi\ 10\ (2, r)$ $b_r = \psi\ 10\ (3, r)$
$1_r =$ ROOT (1) $2_r =$ ROOT (2)
$3_r =$ ROOT (3) $4_r =$ ROOT (4)
$d_r = d_t$
$a_r = 3$
$\neq$
$=$
$29 \leq b \leq 32$
yes
no
root $1_r2_r3_r4_r \in \Delta'$
no
yes
root $1_r2_r3_r4_r \in \Delta$
no
yes
under a_r, b_r, c_r
no
yes
8
yes
$d_r \in d_\Delta$
no
5

Fig. 11. Flow Chart of Test of Lexical Conditions.

In it the following operations are performed:

(1) The values of a_r, b_r, d_r, 1_r, 2_r, 3_r, and 4_r are computed for the given word-form. The values of c_r is already computed.

(2) The value of a_r is examined. If $a_r = 3$, the input word is considered a noun and the value of b_r is examined. If b_r is 29 through 32, dictionary Δ', is interrogated, if not, dictionary Δ is interrogated. If $a_r \neq 3$, dictionary Δ is always interrogated.

(3) The appropriate dictionary is interrogated for root $1_r2_r3_r4_r$. Dictionary Δ requires that every character of the root under test match the corresponding character of the dictionary entry. Dictionary Δ' permits one exception: if a character of the root under test has zero value (*), it is considered to match the corresponding character of the dictionary entry. If no dictionary entry is found, computation ceases for the given value of c_r and control is returned to Block 5.

(4) If the root under test has an entry in the dictionary, the dictionary is further interrogated to determine that the matching entry is of root class c_r and that it has entries under inflection class a_r and stem class b_r. If any of these tests are not satisfied computation ceases and control is returned to Block 5.

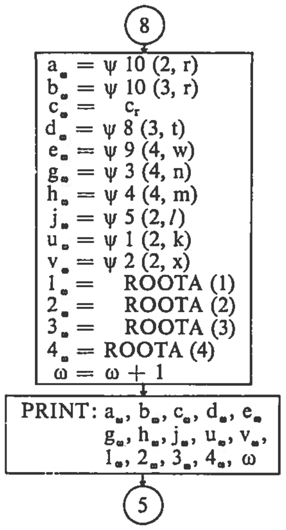
(5) If tests (1) through (4) are satisfied, the value of d_r is compared to the set of values of d listed in the dictionary that are valid for the given entry. If d_r is a member of the specified set, a valid parsing of the input word has been computed, and control is advanced to Block 8. If not, computation ceases for the given value of c_r and control is returned to Block 5. This test is necessary because some words do not appear in all inflections. For example, most nouns have fixed gender, however, some change inflectional gender in the plural. The appendix is a sample Hebrew dictionary containing the data required for this algorithm.

This completes the description of Block 7 of Figure 2.

J. *Block 8: Computation of Grammatical Attributes*

Block 8 of Figure 2 computes the value of ω and grammatical attributes a_ω, b_ω, c_ω, d_ω, e_ω, g_ω, h_ω, u_ω, v_ω, 1_ω, 2_ω, 3_ω, and 4_ω from the appropriate tables and instructs the printing of these values.

Figure 12 is a detailed flow chart of Block 8 of Figure 2.



8
$a_\omega = \psi\ 10\ (2, r)$
$b_\omega = \psi\ 10\ (3, r)$
$c_\omega = c_r$
$d_\omega = \psi\ 8\ (3, t)$
$e_\omega = \psi\ 9\ (4, w)$
$g_\omega = \psi\ 3\ (4, n)$
$h_\omega = \psi\ 4\ (4, m)$
$j_\omega = \psi\ 5\ (2, l)$
$u_\omega = \psi\ 1\ (2, k)$
$v_\omega = \psi\ 2\ (2, x)$
$1_\omega =$ ROOTA (1)
$2_\omega =$ ROOTA (2)
$3_\omega =$ ROOTA (3)
$4_\omega =$ ROOTA (4)
$\omega = \omega + 1$
PRINT: a_ω, b_ω, c_ω, d_ω, e_ω g_ω, h_ω, j_ω, u_ω, v_ω, 1_ω, 2_ω, 3_ω, 4_ω, ω
5

Fig. 12. Flow Chart for Computation of Grammatical Attributes.

The operations are self explanatory. After these computations are complete, control returns to Block 2 of Figure 2 for the generation of a new word-form. This

Table 16

Output of Program

No.	Word	ω	a	b	c	d	e	g	h	j	1234
1	AB	1	3	27	0	10	0	0	0	0	AB**
2	ABYW	1	3	27	0	12	12	0	0	0	AB**
3	ABWTYHM	1	3	27	0	18	35	0	0	0	AB**
4	BYYT	1	3	29	0	10	0	0	0	0	BYYT
5	BYTH	1	3	29	0	12	31	0	0	0	BYYT
6	BTYHN	1	3	29	0	17	36	0	0	0	BYYT
7	OSM	1	1	13	21	10	35	0	0	0	OSS*
		2	1	14	22	10	35	0	0	0	OWS*
		3	2	14	22	12	35	0	0	0	OWS*
		4	1	14	12	10	35	0	0	0	OSH*
8	YSWB	1	1	14	24	20	0	0	0	0	SWB*
		2	1	14	24	20	0	0	0	0	SWB*
		3	2	02	06	10	0	0	0	0	YSB*
		4	2	12	06	20	0	0	0	0	YSB*
		5	2	02	06	12	0	0	0	0	YSB*
9	TSTWBBNH	1	1	10	24	26	0	0	0	0	SWB*
		2	1	10	24	28	0	0	0	0	SWB*
10	YSYBYM	1	1	07	24	20	35	0	0	0	SWB*
		2	1	07	07	20	35	0	0	0	NSB*
11	HTPRNCTY	1	1	06	32	14	0	0	0	0	PRNC
12	AWKLYH	1	1	05	09	33	31	0	0	0	AKL*
		2	1	12	09	24	31	0	0	0	AKL*
		3	1	12	09	32	31	0	0	0	AKL*
		4	2	12	09	17	31	0	0	0	AKL*
		5	3	04	0	17	31	0	0	0	AKL*
13	AKWL	1	1	14	21	24	0	0	0	0	KLL*
		2	2	02	09	10	0	0	0	0	AKL*
		3	2	12	09	20	0	0	0	0	AKL*
		4	2	02	09	12	0	0	0	0	AKL*
		5	2	12	09	21	0	0	0	0	AKL*
14	MLKWT	1	3	32	0	10	0	0	0	0	MLK*
		2	3	01	0	16	0	0	0	0	MLK*
		3	3	01	0	18	0	0	0	0	MLK*
15	MLKWYWTYHN	1	3	32	0	18	36	0	0	0	MLK*
16	PH	1	3	28	0	10	0	0	0	0	PYYH
17	PYK	1	3	28	0	12	20	0	0	0	PYYH
18	PYYWTYHM	1	3	28	0	18	35	0	0	0	PYYH
19	AWKLYHW	1	1	05	09	33	30	0	0	0	AKL*
		2	1	12	09	24	30	0	0	0	AKL*
		3	1	05	09	32	30	0	0	0	AKL*
20	TWAKLWH	1	1	08	09	26	31	0	0	0	AKL*
		2	1	12	09	26	31	0	0	0	AKL*
		3	1	08	09	28	31	0	0	0	AKL*
		4	1	12	09	28	31	0	0	0	AKL*
		5	1	08	09	27	31	0	0	0	AKL*
		6	1	12	09	27	31	0	0	0	AKL*
21	HYLD	1	1	12	06	10	0	0	1	0	YLD*
		2	1	04	06	32	0	0	1	0	YLD*
		3	1	12	06	32	0	0	1	0	YLD*
		4	2	04	06	20	0	1	0	0	YLD*
		5	2	04	06	20	0	0	1	0	YLD*
		6	3	01	0	10	0	0	1	0	YLD*
		7	3	01	0	10	0	1	0	0	YLD*
		8	2	04	06	21	0	0	1	0	YLD*
		9	3	01	0	12	0	1	0	0	YLD*
		10	3	01	0	12	0	0	1	0	YLD*
22	WLKSYWLYKWNW	1	1	07	05	25	15	0	1	1	WLK*

completes the description of the computer program for the algorithm.

V. TEST OF THE ALGORITHM

Twenty-two Hebrew words were manually analyzed by the algorithm as programmed in Section IV. The first twenty-one words were previously used to test the word generating algorithm. The last word is the example given in Section II containing all but one word constituent. Table 16 is the output of the program. The analysis of each word is correct and exhaustive. The informed reader will recognize that these words represent some of the most difficult analytical problems.

Of the twenty-two words, twelve have unique analyses. The multiple analyses result from grammatical, semantical or orthographical ambiguities which are inherent in the language. The eighth word produced two identical analyses. This is due to a redundant entry in Table 10. A careful editing of this table will eliminate redundant analyses. The last word is computed to be the *Hiphil* stem of root WLK; in truth, the root is HLK. This is due to an irregularity of the root in the *Hiphil* stem which substitutes W for H in the orthography. The dictionary lists this word under WLK for the *Hiphil* stem to compensate for this irregularity. There are a few uniquely irregular words such as this that require entries in the dictionary contrary to traditional classification.

There are a few words that the algorithm in its present form will not analyze, such as verbs in the *Nithpael* stem. This is true only because the forms of the words are not presently included in the tables.

An indication of the efficiency of the algorithm is obtained from the number of different combinations of word constituents produced in the analysis of a word. If all possible combinations were produced the number would exceed 22 million. However, using economizing techniques reduced this number to an average of 871 combinations for the twenty-two words analyzed, with a minimum of 362 combinations and a maximum of 1,479 combinations. Table 17 is a summary of the number of operations required to analyze each test word. There was an average of 24.6 dictionary interrogations per word and an average of 2.6 parsings per word.

As far as memory size is concerned, approximately 5,000 words are required for storage of the tables. This could be broken down still further into approximately 15,000 characters. The complete dictionary will require much more space and probably should be stored on tape. In which case, the program should be divided into two sections. The first section would consist of every thing before the dictionary interrogation, in which section a list of all dictionary inquiries is made. The second section would consist of sorting the inquiries in alphabetic order, one pass on the dictionary, and sorting the resultant analyses back into their original order for output. With this type of approach, smaller machines can be used for the program. It is concluded that the algorithm is a successful approach to automatic parsing of Hebrew words and that with further expasion of the tables, it will analyze any Hebrew word.

TABLE 17
Summary of Number of Operations

Word Number	Number of Combinations	Number of Dictionary Interrogations	Number of Analyses
1	362	28	1
2	1096	39	1
3	489	15	1
4	1188	48	1
5	1317	76	1
6	893	22	1
7	681	32	4
8	804	24	5
9	459	8	2
10	1166	13	2
11	849	1	1
12	1295	15	5
13	804	28	5
14	1479	49	3
15	491	7	1
16	471	31	1
17	876	41	1
18	410	4	1
19	872	12	3
20	995	13	6
21	1269	36	10
22	898	3	1
Total	19,164	541	57
Average	871	24.6	2.6

APPENDIX
Sample Hebrew Dictionary

This Appendix is a sample Hebrew dictionary with the data required by the Algorithm for Analyzing Hebrew Words. It contains the words needed in Section V for the test of the algorithm. The columns list the following data: root letters (column 1), inflection class (a) (column 2), stem class (b) (column 3), root class (c) (column 4), "logeme" class (d) (column 5), syntactic Number and Gender (G) (column 6), syntactic function (S) (column 7), and English meaning (column 8).

The fifth column contains the "logeme" classes (d) which are valid for the given word. The symbols in this column are as follows:

O — all values of d are valid
M — masculine, valid for d = 10, 12, 15, 17
MS — masculine singular, valid for d = 10, 12
MP — masculine plural, valid for d = 15, 17
F — feminine, valid for d = 11, 13, 16, 18
FS — feminine singular, valid for d = 11, 13
FP — femine plural, valid for d = 16, 18.

The sixth column contains the syntactic number and gender (G) of the given word, that is, the number and gender of the word in syntactic constructions. For example, the word AB (father) takes the feminine plural inflection ABWT (fathers) but it governs masculine plural verbs and adjectives in syntactic constructions. The symbols in this column are the same as in column five except 0 indicates that syntactic number and gender are the same as the inflectional number and gender.

The seventh column gives the syntactic function (S). The symbols are as follows: N — noun, V — verbal, A — adjective, D — adverb. Many other syntactic functions exist but there are no examples in the sample dictionary.

Sample Hebrew Dictionary Δ

Root	a	b	c	d	G	S	Meaning
ABB*	1-2	07	21	0	0	V	to bring forth shoots
	3	03	0	M	M	N	flute
ABH*	1-2	13	11	0	0	V	to want
	3	05	0	M	M	N	reed
AWB*	3	01	0	MS	MS	N	necromancy
	3	01	0	FP	MP	N	necromancy
AYB*	1-2	12	27	0	0	V	to be hostile to
	3	01	0	F	F	N	hatred
	3	04	0	M	M	N	enemy
AKL*	1-2	03	09	0	0	V	to be eaten
	1-2	04	09	0	0	V	to consume
	1-2	05	09	0	0	V	to be consumed
	1-2	07	09	0	0	V	to feed
	1-2	08	09	0	0	V	to be fed
	1-2	12	09	0	0	V	to eat
	3	01	0	F	F	N	food
	3	01	0	0	0	A	burnt
	3	04	0	M	M	N	food
	3	05	0	M	M	N	cancer
	3	22	0	M	M	N	eater
BYT*	3	01	0	MS	FS	N	*Beth*, second letter
	3	01	0	FP	FP	N	*Beth*, second letter
BT**	3	25	0	F	F	N	wasteland

Root	a	b	c	d	G	S	Meaning
HLK*	1-2	03	0	0	0	V	to be gone
	1-2	04	0	0	0	V	to walk about
	1-2	06	0	0	0	V	to walk to and fro
	1-2	14	0	0	0	V	to walk
	3	01	0	M	M	N	mood
	3	01	0	F	F	N	rule, practice
	3	02	0	M	M	N	step
	3	02	0	F	F	N	walking
	3	05	0	M	M	N	traveler
WLK*	1-2	07	05	0	0	V	to lead
YLD*	1-2	03	06	0	0	V	to be born
	1-2	04	06	0	0	V	to assist in birth
	1-2	05	06	0	0	V	to be born
	1-2	06	06	0	0	V	to declare one's pedigree
	1-2	07	06	0	0	V	to cause to bear
	1-2	08	06	0	0	V	to be born
	1-2	12	06	0	0	V	to beget
	3	01	0	M	M	N	boy, child
	3	01	0	F	F	N	girl
	3	02	0	M	M	N	native
	3	03	0	M	M	N	child
	3	06	0	0	0	A	born
YSB*	1-2	03	06	0	0	V	to be inhabited
	1-2	04	06	0	0	V	to set, place
	1-2	05	06	0	0	V	to be seated
	1-2	06	06	0	0	V	to settle oneself
	1-2	07	06	0	0	V	to cause to sit
	1-2	08	06	0	0	V	to be made to dwell
	1-2	12	06	0	0	V	to sit
	3	06	0	M	M	N	settlement
KYL*	1-2	07	27	0	0	V	to hold
	1-2	13	27	0	0	V	to comprehend
	3	01	0	M	M	N	to measure
KLL*	1-2	03	21	0	0	V	to be included
	1-2	04	21	0	0	V	to crown
	1-2	05	21	0	0	V	to be done with perfection
	1-2	06	21	0	0	V	to crown oneself
	1-2	07	21	0	0	V	to perfect
	1-2	14	21	0	0	V	to complete
	3	01	0	M	M	N	principle
	3	04	0	M	M	N	community
MLK*	1-2	03	0	0	0	V	to consider
	1-2	06	0	0	0	V	to make oneself king
	1-2	07	0	0	0	V	to cause to reign
MLK*	1-2	08	0	0	0	V	to be made king
	1-2	14	0	0	0	V	to reign
	3	01	0	M	M	N	king

Root	a	b	c	d	G	S	Meaning
	3	01	0	F	F	N	queen
NKL*	1-2	04	07	0	0	V	to beguile
	1-2	06	07	0	0	V	to conspire
	1-2	14	07	0	0	V	to deceive
	3	05	0	M	M	N	deceit
NP**	3	25	0	F	F	N	sieve
NPH*	1-2	04	10	0	0	V	to sift
	1-2	05	10	0	0	V	to be sifted
NSB*	1-2	04	07	0	0	V	to cause to blow
	1-2	07	07	0	0	V	to drive away
	1-2	14	07	0	0	V	to blow
	3	01	0	M	M	N	blowing
	3	03	0	F	F	N	chaff
OWS*	1-2	14	22	0	0	V	to hurry
OYS*	3	02	0	MS	F	N	The Great Bear
OSH*	1-2	03	12	0	0	V	to be done/made
	1-2	04	12	0	0	V	to press, squeeze
	1-2	05	12	0	0	V	to be made
	1-2	07	12	0	0	V	to cause to do/make
	1-2	14	12	0	0	V	to do/make
OSH*	3	04	0	M	M	N	maker
OSS*	1-2	04	21	0	0	V	to dim/darken
	1-2	06	21	0	0	V	to exercise ones strength
	1-2	13	21	0	0	V	to waste away
PWK*	3	01	0	M	M	N	eye-paint
PRNC	1-2	04	32	0	0	V	to support
	1-2	06	32	0	0	V	to support oneself
	3	19	0	M	M	N	manager
	3	19	0	F	F	N	provision

Root	a	b	c	d	G	S	Meaning
SBB*	1-2	04	18	0	0	V	to chip/chisel
	3	01	0	M	M	N	splinter
	3	04	0	0	0	A	apostate
SWB*	1-2	04	22	0	0	V	to lead back
	1-2	07	22	0	0	V	to bring back
	1-2	08	22	0	0	V	to be brought back
	1-2	09	22	0	0	V	to lead back
	1-2	10	22	0	0	V	to backslide
	1-2	14	22	0	0	V	to return
	3	01	0	0	0	D	again
	3	01	0	F	F	N	retirement
SYB*	3	01	0	F	F	N	sojourning

Sample Hebrew Dictionary Δ′

AB**	3	27	0	MS	MS	N	father
	3	27	0	FP	MP	N	father
ABB*	3	31	0	F	F	N	ague
AKL*	3	32	0	F	F	N	voracity
BYYT	3	29	0	M	M	N	house
YLD*	3	32	0	F	F	N	childhood
KYL*	3	32	0	F	F	N	avarice
KLL*	3	32	0	F	F	N	totality
MLK*	3	32	0	F	F	N	kingdom
OSS*	3	31	0	MS	FS	N	lamp
	3	31	0	FP	FP	N	lamp
PYYH	3	28	0	M	M	N	mouth
SBB*	3	32	0	F	F	N	wilderness

May 1968

THE LAW OF SOFTWARE – SECOND ANNUAL CONFERENCE

The Law of Software, Second Annual Conference, sponsored by the Computers-in-Law Institute of George Washington University, was held in Washington, D.C., October 8-9, 1969. The purpose of the conference was to explore the recent legal developments affecting the computer software industry. The traditional methods of protecting intellectual property, i.e. copyright, patent, and trade secrets, were reviewed and the shortcomings and problems inherent in each discussed. A summary of the highlights of this part of the program follows.

James W. Falk, Patent Attorney-Director, Bell Telephone Laboratories, spoke on copyright infringement problems. In his opinion copyright is not the way to obtain protection and he suggested that courts may hold that an attempt to use a copyright to prevent use of a program is an abuse of copyright. His opinion is based on the thesis that there is a line drawn between patents and copyrights and there is no overlap in the protection afforded by these two schemes. Something that is explicable is copyrightable, whereas something that is functional is patentable. The same item may have characteristics which are both explicative and functional. It may be both copyrighted and patented, but the copyright cannot extend to cover the functional aspects, and the patent cannot cover the explicative aspects. There is a patent doctrine that printed matter is not patentable. However, where the printed matter is not explicative but structural, it comes within the patent system. For example, the Court of Customs and Patent Appeals in *In Re Jones*, 373 F.2d 1007 (C.C.P.A. 1967), upheld the patentability of a code disc.

There have been no decisions involving computer programs in which the question of what constitutes an infringement has been raised. For this reason Mr. Falk used analogies to support his thesis of the distinction in the explicative and functional. Building plans, for example, have been held to be copyrightable and the copyright is infringed if copies are made of the plans. If, however, someone builds a house in accordance with the copyrighted plans, there is no infringement because the copyright does not cover the practice of the art explained in the copyrightable materials. Other cases have held that the copyright law does not afford protection to ideas expressed in copyrightable materials. Among these cases are *Downes* v. *Culbertson*, 153 Misc. 14, 275 N.Y.S. 233 (Sup. Ct. 1934), in which a system of contract bridge playing was held not to be copyrightable, and the leading case, *Selden* v. *Baker*, 101 U.S. 99 (1879). In this case one Selden had obtained a copyright for his book entitled *Selden's Condensed Ledger, or Bookkeeping Simplified*, the object of which was to exhibit and explain a peculiar system of book-keeping. The book consisted of an introductory essay explaining the system to which was annexed certain forms consisting of ruled lines and headings illustrating the system. The Supreme Court held that blank account books were not the subject of copyright and that the copyright of Selden's book did not confer upon him the exclusive right to make and use account books ruled and arranged as designed by him. In explaining the difference between that which may be protected by copyright and that which is the subject of patents, the Court stated: "The object of the one is explanation. The object of the other is use. The former may be secured by copyright. The latter can only be secured, if it can be secured at all, by letters patent."

These decisions lead Mr. Falk to the conclusion that a copyright may protect a software program from being copied because it is explicative, but when the program is used to control a computer there is no infringement of copyright.

Morton D. Goldberg of the New York firm of Schwab & Goldberg, commenting on the scope of the rights under copyright, expressed the opinion that incentives are necessary to promote the progress of the software art and that law should advance as well as technology. He questioned the applicability of old copyright decisions to the present situation, asserting that computer programs differ so much from the facts in these cases as to be a difference in kind.

He cited the leading case, *Selden* v. *Baker*, as presenting a significant obstacle to securing the scope of exclusive rights which would make the copyright of a computer program truly meaningful. But he stated that he believed the obstacle could be overcome. One argument that he used was that the decisions which were relied on in *Selden* v. *Baker* for authority have long since been rejected in the light of the expanded scope of protection which Congress and courts now afford to copyrighted works. At any rate, Mr. Goldberg was of the opinion that even if Selden's bookkeeping system could be used by Mr. Baker to prepare his own accounts by hand, a million-dollar computer system should not be available for use by the public without the owner's consent.

Attention was invited by Mr. Goldberg to the Standard Reference Data Act passed by Congress in 1968. This Act directs the Secretary of Commerce not only to compile and disseminate standard reference data, but also to secure copyright on such data on behalf of the government. Senate and House reports on this bill indicate that members of Congress were aware that a large part of this data will be disseminated in the form of computer programs and data files embodied on computer tapes. While admitting that this was not a legislative determination of the protection of programs, Mr. Goldberg pointed out that the Act does embody a Congressional determination that the principles of the Copyright Act apply to protect at least the government's programs and data bases.

The use of common law copyright was also discussed as a means of protection of computer programs. The federal preemption of the copyright area does not extend to unpublished works. Mr. Goldberg suggested the possibility that works which cannot secure statutory copyright may be protectible under common law copyright because the scope of the latter is broader in some respects. A problem here, however, is the danger that the program may be "published". This would divest the common law copyright proprietor of his right. Mr. Goldberg was of the opinion that it is possible to restrict a licensee's exercise of intangible rights in a program to the extent that it might not be deemed a "publication". The IBM license agreement appeared to him to be an agreement of this type.

The pending copyright legislation, Senate bill 51, 91st Congress, 1st Session, does provide for the copyright of computer programs, but the scope of the rights such programs will be accorded is not clear. The bill provides for the establishment of a National Commission on New Technological Uses of Copyrighted Works which will study the problem. The bill, however, does not provide for the membership of software producers on this commission. IBM has proposed the establishment of a registration system which would require public disclosure of a description of the concepts of the program but would preserve the secrecy of the program itself. Liability would be incurred for copying, but not for independent creation.

David Bender of the Computers-in-Law Institute at George Washington University discussed the applicability of the state common law of trade secrets. Because of the uncertainties of patent and copyright protection, trade secret protection has seemed, until recently, to be a safer route for software proprietors to follow in some instances. The Restatement of Torts defines a trade secret as consisting of "any formula, pattern, device, or compilation of information which is used in one's business, and gives one an opportunity to obtain an advantage over competitors who do not know or use it". A computer program, according to Mr. Bender, fits this definition. The historical basis for allowing trade secret protection lies in the fundamental notion of fairness which permits the owner of a trade secret to keep the work which he has done, or paid for doing, for himself.

Now, however, the decision in *Lear, Inc.* v. *Adkins*, 395 U.S. 653 (1969), brings this method of protection into question. In this case Justice Black, writing a separate opinion for himself and two other members of the Court, stated in reference to the contractual right of the inventor to license his invention while his patent application was pending, as follows:

> One who makes a discovery may, of course, keep it secret if he wishes, but private arrangements under which self-styled 'inventors' do not keep their discoveries secret, but rather disclose them, in return for contractual payments, run counter to the plan of our patent laws, which tightly regulate the kind of inventions that may be protected and the manner in which they may be protected. The national policy expressed in the patent laws, favoring free competition and narrowly limiting monopoly, cannot be frustrated by private agreements among individuals with or without the approval of the State.

Mr. Bender pointed out that legislation could possibly preserve the common law of trade secrets and that appropriate legislation is now before Congress.

Virgil E. Woodcock of the Philadelphia firm of Woodcock, Phelan & Washburn, spoke on the implications of the recent landmark decision *In Re Prater and Wei*, 162 U.S.P.Q. 541 (C.C.P.A., 1969). Mr. Woodcock argued on behalf of the applicant in this case in the rehearing before the Court of Customs and Patent Appeals. As an introduction, he began by quoting the guidelines issued by the Patent Office on October 22, 1968, as follows:

> Special problems of patentability arise in the computer and data processing field revolving around logical processes and mathematical equations. Mental processes may not be patented although they may be of enormous importance; *In re Abrams*,

1951 C.D. 264, 38 CCPA 945, 89 USPQ 266. A process or method is directed to patentable subject matter only if it is performed on physical materials and produces some appreciable change in their character or condition; *In re Shao Wen Yuan*, 1951 C.D. 286, 38 CCPA 967, 89 USPQ 324; *Cochrane* v. *Deener*, 94 U.S. 780, 1877 C.D. 242. Accordingly, a computer programming process which produces no more than a numerical, statistical or other informational result is not directed to patentable subject matter. Such a process may, however, form a part of a patentable invention if it is combined in an unobvious manner with physical steps of the character above referred to as, for example, in the knitting of a pattern or the shaping of metal.

The decision in *Prater and Wei* makes it clear, according to Mr. Woodcock, that the Patent Office was wrong in its limited interpretation of the cases cited to support the guidelines. *Prater and Wei* concerns an application for a patent on an invention which includes both a method and an apparatus for the processing of conventionally obtained spectrographic data to produce a quantitative spectrographic analysis of a qualitatively-known mixture. The applicants contended that they had invented a machine which made possible the utilization of their discovery without human intervention and that their process could be practiced by a properly programmed general-purpose computer. The Court, in overruling its earlier decision rejecting the application, held that the decisions in *Abrams*, *Yuan*, and *Cochrane* v. *Deener* were not controlling. Both *Abrams* and *Yuan* were concerned with purely mental steps which do not form a process which falls within the scope of patentability as defined by the statute. The Court also stated that the decision in *Cochrane* v. *Deener* has sometimes been misconstrued as a rule requiring all processes, in order to be patentable, to operate physically upon substances. According to the court, it was not the intention to limit patentability, but to point out that a process is not limited to the means used in performing it. While this decision does not answer all the questions in this area, footnote 29 of the opinion, Mr. Woodcock pointed out, should give great comfort to those who are seeking to obtain patent protection for new and unobvious operations of digital computers. This footnote says:

No reason is now apparent to us why, based on the Constitution, statute, or case law, apparatus *and* process claims broad enough to encompass the operation of a programmed general-purpose digital computer are necessarily unpatentable. In one sense, a general-purpose digital computer may be regarded as but a storeroom of parts and/or electrical components. But once a program has been introduced, the general-purpose digital computer becomes a special-purpose digital computer (i.e., a specific electrical circuit with or without electro-mechanical components) which, along with the processes by which it operates, may be patented, subject, of course, to the requirements of novelty, utility, and nonobviousness. Based on the present law, we see no other reasonable conclusion.

William E. Schuyler, Jr., the Commissioner of Patents, announced that as a result of the *Prater and Wei* decision, the Patent Office guidelines for computer programs were being rescinded. From now on, patent applications for computer programs will be considered on the basis of the merits of the specific inventions sought to be protected. He predicted that the law of patentability of computer programs must develop on a case by case basis, pointing out that many computer programs will be unpatentable because they are obvious to a skilled programmer. He mentioned that the Copyright Office has prescribed unique regulations for registration of computer programs, but admitted that lawyers in the field are concerned about the scope of protection afforded by the copyright law. In his opinion a new form of protection is necessary for the software industry. For this reason the National Council of the Patent Law Association is making a study of the problems involved in order to propose legislation to Congress.

Donald I. Baker, of the Antitrust Division of the Department of Justice, speaking on the antitrust implications of software protection, expressed the opinion that a decision to grant a legal monopoly should rest on economic need. He said that in this, the age of computers, finding the right answer is vitally important — far too important to be left to lawyers and their medieval legal analogies to existing types of patent or copyright protection.

As these discussions indicate, the state of the law concerning the protection of software is still uncertain. The law should protect the software buyer from a monopoly situation where the software producer, in the absence of competition, can charge unreasonable prices. The software seller needs protection against imitators who take his program without regard for his investment. Perhaps as Commissioner Schuyler suggests, new legislation is the only solution.

University of North Carolina FRANCES H. HALL

Contents

Some articles for forthcoming issues:

David R. Barstow, TIRADE-II, EDARIT-II: Linguistic Studies on an IBM 1620

H. Eugene Byars, An Investigation into Programming Style

Dolores M. Burton, Aspects of Word Order in Two Plays of Shakespeare

Margaret Scanlon Cabaniss, Using the Computer for Text Collation

John J. Dreher, Elaine L. Young, Robert E. Norton and John T. Ma, Power Spectral Densities of Literary Rhythms (Chinese)

Richard E. Joiner, Gregorian Chant

Sharon A. Jamieson, Style in Architecture

Gerd Rechten, Pattern Recognition in Character Strings

O. Romayne Smith, Jr., GENDEX: General Indexer of Words with Context — A Concordance Generator