AN ALGORITHM FOR THE AUTOMATIC PHONETIC TRANSCRIPTION OF RUSSIAN

by 5408

MICHAEL JOHN McCORMICK

B. A., Kansas State University, 1969

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF ARTS

Department of Modern Languages

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1971

Approved by:

William A. Coates
Major Professor

CONTENTS

LIST OF PLATES

The purpose of this paper is to describe a working algorithm for the automatic phonetic transcription of Russian. An algorithm is defined as "a mechanical procedure which, when applied to a certain class of symbols (input symbols), ultimately produces an output symbol."[1] Characteristics essential to the concept of an algorithm include the following:

a) an algorithm is a set of instructions of finite size;

b) an operator responds to the instructions and carries out the computation;

c) certain devices allow the various steps of the computation to be executed, stored and brought back again;

d) The procedures are essentially discrete;

e) The set of elementary operations to be carried out is perfectly determined (deterministic): at each step there is just one possible way of going on to the next step.[2]

As Gross and Lentin point out in their Introduction to Formal Grammars, the analogy to the electronic computer is clear:

a) corresponds to the program;

[1] M. Gross and A. Lentin, Introduction to Formal Grammars, trans. M. Salkoff (New York, 1970), p. 43.

[2] Gross and Lentin, p. 44, citing H. Rogers, Jr., "Recursive Functions and Effective Computability" (mimeographed, Dept. of Mathematics, Massachusetts Institute of Technology, 1961).

b) corresponds to the electronic computer in which the program is executed;

c) corresponds to the computer's memory;

d) corresponds to the numerical character of the computer;

e) corresponds to the mechanical, deterministic character of the machine execution of the program.[3]

Our algorithm is written particularly for use on an electronic computer, but has value in itself as a logical algorithm stated in explicit, formal terms, and as such as a functioning part of a formal grammar. This algorithm is expressed in the PL/1 computer programming language, a formal metalanguage which provides us with a means of addressing the operator, in this case the IBM 360 computer. At the same time, the algorithm is written following the generative-transformational theory of phonology, and as such provides a working model which reflects computationally each stage of that phonological theory.

[3] Gross and Lentin, p. 44.

THE PHONOLOGICAL THEORY

The phonological theory followed by this algorithm is that provided by the transformational-generative model, as developed by Morris Halle, Noam Chomsky and Robert Harms,[4] among others. The basis for the actual rules was taken largely from Halle's work, The Sound Pattern of Russian. This work is from an early stage of the generative theory, and the rules it contains are stated in a less explicit form than was desired, but we have translated them into more formal terms, for the consonants, and constructed our own rules for vowels.

The Distinctive Features

The phonological theory has at its base a system of phonological feature specifications, which as originally formulated by Roman Jakobson and others refer to physically measurable acoustic properties of utterances.[5] Among the features listed by Jakobson were the following: consonantal, vocalic, compact and diffuse, grave and acute, flat and sharp, continuant, strident, tense, voiced and nasal. These features parallel

[4] Morris Halle, The Sound Pattern of Russian (The Hague, 1959).

Noam Chomsky and Morris Halle, The Sound Pattern of English (New York, 1968).

Robert T. Harms, Introduction to Phonological Theory (Englewood Cliffs, New Jersey, 1968).

[5] Roman Jakobson and Morris Halle, Fundamentals of Language (The Hague, 1956).

Roman Jakobson, Gunnar Fant and Morris Halle, Preliminaries to Speech Analysis (Cambridge, Massachusetts, 1963).

for the most part the articulatory phonetic specifications of the International Phonetic Alphabet. One primary advantage to Jakobson's system of phonetic description is that the same features are used to describe both consonants and vowels, while the IPA system uses different terms for vowels than for consonants. One difficulty with Jakobson's features is that while the same feature specifications or labels are used for both vowels and consonants, in a few cases they must be interpreted differently for vowels than for consonants. The opposition compact vs. diffuse, for example, is interpreted as low vs. high for vowels and as back vs. front for consonants, or essentially a ninety degree shift in orientation. The opposition grave vs. acute is to be interpreted as back vs. front for vowels and as peripheral vs. central for consonants, where peripheral refers to both extreme front and back, as opposed to centrally articulated consonants.

As Harms points out, these Jakobsonian feature specifications are not absolute but relative in classifying a group of sounds. The four-vowel system /i e o a/ would most likely be analyzed as the following:

	i	e	o	a
grave	–	–	+	+
compact	–	+	–	+

where /e/ is described as ⟨+compact⟩ or as a low vowel. These classification decisions are based on questions of symmetry and economy of features.[6]

[6] Harms, p. 27.

The Distinctive Feature Matrix

The phonological segments with their feature specifications are generally presented in a matrix display in which the rows represent feature specifications labeled at the left, and the columns are headed by individual phonological segments. These segments might be referred to as morphophonemes, although the generative phonologists generally avoid this term in favor of something like 'underlying phonological form', as the definition and function of these segments throughout the generative system as a whole forbid their being identified with the structuralist concepts.

The basic matrix display is the distinctive feature matrix, in which only those features are specified which are required to distinguish each segment from every other segment. In the distinctive feature matrix the specifications are binary, that is, plus or minus. A segment is described as characterized by a particular feature (+) or not characterized by a particular feature (-). If a particular feature is not distinctive for a segment, then a zero (0) is entered in the column dominated by that segment. This is the nature of the feature matrix as it appears upon entry into the phonological component.

The inclusion of both parameters of feature oppositions such as compact vs. diffuse would depend on the number of distinctions to be made. For example, with the opposition high vs. low, if one wished to distinguish only two horizontal levels, only would be required.

1) $[+\text{high}]$

2) $[-\text{high}]$

If three levels were to be distinguished, the opposite feature would be required in combination:

1) $\begin{bmatrix}+\text{high}\\-\text{low}\end{bmatrix}$

2) $\begin{bmatrix}-\text{high}\\-\text{low}\end{bmatrix}$

3) $\begin{bmatrix}-\text{high}\\+\text{low}\end{bmatrix}$

With two parameters, of course, four combinations are mathematically possible, although the combination $\begin{bmatrix}+\text{high}\\+\text{low}\end{bmatrix}$ must be excluded for obvious physical reasons. A fourth level would require some sort of mid feature used in combination with the others.

In *The Sound Pattern of Russian*, Halle uses the basic Jakobsonian features described above to classify the phonological segments of Russian. His distinctive feature matrix is illustrated in plate I.[7]

In his more recent work, particularly in *The Sound Pattern of English*, with Noam Chomsky, Halle modifies the feature system slightly, replacing some of the acoustics-oriented features with articulatory terms, and changing some interpretations. Thus for vowels "diffuse" becomes "high", "compact" becomes "low", "grave" becomes "back", and "acute" becomes "non-back". For consonants, "diffuse" becomes "anterior", to describe consonants articulated in front of the palatoalveolar region, and "compact" becomes

[7] Halle, p. 45.

PLATE I

Halle's Morphophonemes of Russian

	j	t	d	t,	d,	n	n,	c	s	z	s,	z,	p	b	p,	b,	m	m,	f	v	f,	v,	č	š	ž	k	k,	g	x	e	'e	o	'o	a	'a	i	'i	u	'u	r	r,	l	l,
VOCALIC	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+
CONSONANTAL	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	−	−	−	−	−	+	+	+	+
DIFFUSE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	−	−	−	−	−	+	+	+	+	0	0	0	0
COMPACT	0	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+	−	−	−	−	+	+	0	0	0	0	0	0	0	0
LOW TONALITY	0	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+	+	+	+	−	−	−	+	+	+	+	−	−	+	+	0	0	−	−	+	+	0	0	0	0
STRIDENT	0	−	−	−	−	−	−	+	+	+	+	+	−	−	−	−	−	−	+	+	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
NASAL	0	−	−	−	−	+	+	0	0	0	0	0	−	−	−	−	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CONTINUANT	0	0	0	0	0	0	0	−	+	+	+	+	0	0	0	0	0	0	0	0	0	0	−	+	+	−	−	−	+	0	0	0	0	0	0	0	0	0	0	−	−	+	+
VOICED	0	−	+	−	+	0	0	0	−	+	−	+	−	+	−	+	0	0	−	+	−	+	0	−	+	−	−	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SHARPED	0	−	−	+	+	−	+	0	−	−	+	+	−	−	+	+	−	+	−	−	+	+	0	0	0	−	+	0	0	0	0	0	0	0	0	0	0	0	0	−	+	−	+
ACCENTED	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	+	−	+	−	+	−	+	−	+	0	0	0	0

"non-anterior". "Grave" becomes "coronal" for consonants to describe consonants articulated with the blade of the tongue raised from its neutral position. "Acute" for consonants becomes "non-coronal" to describe the labials and consonants articulated with the body of the tongue.

We have chosen to use Halle's later features in this algorithm, and we have adapted his distinctive feature matrix to accomodate the new terms. The revised matrix is given in Plate II. In order to reduce machine computation time we have chosen to eliminate the duplication of columns for sharped (palatalized) counterparts, since sharping is represented as a feature. We include sharping with stress at the bottom of the matrix, left unspecified for those segments where it can be of variable application, that is, where the application is to be specified in the context-sensitive rules to follow, and which apply to actual utterances.

The Blank-filling Rules and the Context-sensitive Rules

The phonological computation begins with what are called "blank-filling" or "context-free redundancy" rules which replace the zeros of the matrix with either a plus or a minus. The classificatory matrix is no longer a distinctive feature matrix but rather a completely specified matrix.

These rules are followed by more ordered rules, the context-sensitive rules, which apply to the sequences of segments as they appear in utterances and alter them according to their phonetic environment. The result, at the lowest level of phonetic output, is an utterance in phonetic transcription with completely specified features.

PLATE II

Morphophonemes of Russian

	j	t	d	n	c	s	z	p	b	m	f	v	č	š	ž	k	g	x	e	o	a	i	u	r	l
VOCALIC	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	+	+	+	+	+	+	+
CONSONANTAL	-	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	+	+
ANTERIOR	0	+	+	+	+	+	+	+	+	+	+	+	-	-	-	-	-	-	0	0	0	0	0	-	+
CORONAL	0	+	+	+	+	+	+	-	-	-	-	-	+	+	+	-	-	-	0	0	0	0	0	0	0
HIGH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	-	+	+	0	0
LOW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	-	+	0	0	0	0
BACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-	+	0	-	+	0	0
STRIDENT	0	-	-	-	+	+	+	-	-	-	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0
NASAL	0	-	-	+	0	0	0	-	-	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CONTINUANT	0	0	0	0	-	+	+	0	0	0	0	0	-	+	+	-	-	+	0	0	0	0	0	0	0
VOICED	0	-	+	0	0	-	+	-	+	0	-	+	0	-	+	-	+	0	0	0	0	0	0	0	0
ROUND	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SHARP	+				-								+	-	-										
STRESSED																			+	+		+			

THE ALGORITHM

A Simple Prototype

Input

In our computational model we first fill a matrix labeled MATRIX with the distinctive feature specifications of the Russian segment inventory as proposed by Halle. This information is given as data. To explain the general computational approach we have devised, we shall illustrate the algorithm with a trivial example. Consider a morphophoneme inventory of /k, g, s, z, a/. MATRIX in its distinctive feature form is read in as the following:

	K	G	S	Z	A
CONSONANTAL	+	+	+	+	-
VOCALIC	-	-	-	-	+
CONTINUANT	-	-	+	+	0
VOICED	-	+	-	+	0

For our algorithm we shall consider the segments K, G, S, Z, A as the top row or first row of the matrix and not as labels, while the feature names at left are considered labels. Column 1 of the matrix then consists of K+---.

To refer to a specific location in a matrix we use the form F(m,n) where F is the name of the matrix, here MATRIX; and m is the number of the row (numbered from top to bottom) and n the number of the column (numbered from left to right) which intersect to define the location. Thus MATRIX(1,3) refers to the location at the intersection of the first row and the third column, or "S" in the matrix so far defined. These matrices are stored in the computer's memory as arrays and each location on the matrix is addressable in the above form.

Computation

Next the unspecified locations (those filled by 0) are specified by the application of the blank-filling rules. These rules must state the generality assumed for this hypothetical language that all vowels are voiced and continuant. As true vowels are defined by the combination of $\begin{matrix}\text{-consonantal}\\ \text{+vocalic}\end{matrix}$ the phonological rule would take the form

$$\begin{bmatrix}\text{-consonantal}\\ \text{+vocalic}\\ \\ \end{bmatrix} \longrightarrow \begin{bmatrix}\text{------}\\ \text{------}\\ \text{+continuant}\\ \text{+voiced}\end{bmatrix}$$

Because of the nature of the machine, we must ask the computer to consider one item at a time. Therefore in specifying the unspecified locations we ask the machine to consider one column of the matrix at a time, moving from left to right. As it pauses at each column we must tell the machine to determine whether the position in the second row is filled by a '-' and the position in the third row by a '+'. If this trial situation proves true, then we must tell the machine to erase or 'clobber' what is in the position in the fourth row and replace it by a '+', and to do the same to the position in the fifth row. In the formal terms of the PL/1 programming language these instructions take the following form:

```
DO J = 1 TO 5;
IF MATRIX(2,J) = '-' & MATRIX(3,J) = '+' THEN
     DO;
        MATRIX(4,J) = '+';
        MATRIX(5,J) = '+';
     END;
END;
```

where the DO-END link is called a DO-loop if an index is incremented, as here J goes from 1 to 5. This means that this series is executed five times, once for each value of J. The inside DO-END link is called a DO-block and merely includes the operations to be executed if the conditions specified in the preceding line are true. If these conditions prove false, the machine will jump to the last END statement, incrementing J and repeating the computation if J has not reached the value 5.

With a completely specified matrix we must next consider an utterance. This is read into a one-dimensional array named CARD. We shall use the utterance "KAGS" as an example, where the orthographic symbols are the underlying phonological forms in some hypothetical language. We must next define a new matrix which we shall call FEATURE. We have the machine copy the sequence of segments from CARD into the top row of the matrix FEATURE. Then by comparing each segment of FEATURE with each item of the first row of MATRIX, the machine copies the remainder of each column (rows two through five) from MATRIX and places that under the corresponding segment of FEATURE (filling rows two through five of each column of FEATURE). Thus we have FEATURE as follows:

	K	A	G	S
CONSONANTAL	+	−	+	+
VOCALIC	−	+	−	−
CONTINUANT	−	+	−	+
VOICED	−	+	+	−

Next we must apply the context-sensitive rules to the utterance, at this point to the matrix FEATURE. In our phonology of this hypothetical language we shall assume only one context-sensitive rule--that in a cluster of two consonants if the second is voiceless the first is voiceless. In the transformational-generative model this is stated as

$$\begin{bmatrix}\text{+consonantal}\\ \text{-vocalic}\end{bmatrix} \rightarrow \begin{bmatrix}\text{------}\\ \text{------}\\ \text{-voiced}\end{bmatrix} \Big/ \underline{\quad\quad} \begin{bmatrix}\text{+consonantal}\\ \text{-vocalic}\\ \text{-voiced}\end{bmatrix}$$

where '/' is read "in the environment" and '____' represents the position in that environment of the segment on the left of the arrow.

In the PL/1 language we must give the following instructions:

```
  DO J = 1 TO 4;
  IF FEATURE(2,J) ¬= '+' & FEATURE(3,J) ¬= '-' THEN GO TO H;
  IF FEATURE(2,J+1) ¬= '+' & FEATURE(3,J+1) ¬= '-' THEN GO TO H;
  IF FEATURE(5,J+1) = '-' THEN FEATURE(5,J) = '-';
H:END;
```

where ¬= is read "is not equal to".

As the machine completes this computation, FEATURE takes the following form:

	K	A	G	S
CONSONANTAL	+	−	+	+
VOCALIC	−	+	−	−
CONTINUANT	−	+	−	+
VOICED	−	+	−	−

We then ask the machine to consider rows two through five of the first column of FEATURE and compare their contents with the contents of rows two through five of each column of MATRIX. If the machine finds a discrepancy in comparison with a column of MATRIX it moves to the next column of MATRIX. If it finds complete agreement we instruct the machine to copy the contents of the first row position of whatever column of MATRIX it is considering into the first row position of column one of FEATURE. The procedure is repeated for each column of FEATURE.

Output

In final form, or output form, then, FEATURE is the following:

	K	A	K	S
CONSONANTAL	+	-	+	+
VOCALIC	-	+	-	-
CONTINUANT	-	+	-	+
VOICED	-	+	-	-

In PL/1 formal terms this computation takes the form of three nested DO-loops where J is the index for the column of FEATURE; K the index for the column of MATRIX; and I the index for the rows of both FEATURE and MATRIX:

```
DO J = 1 TO 4;
    DO K = 1 TO 5;
        DO I = 2 TO 5;
        IF FEATURE(I,J)¬= MATRIX(I,K) THEN GO TO Q;
        END;
        FEATURE(1,J) = MATRIX(1,K);
    Q:END;
END;
```

For the Russian language, of course, the details of the rules are much more complicated, as we shall see, but the basic operations of matrix comparison and copying remain the same.

The PL/1 Program

In describing the complete algorithm for Russian we shall refer directly to statements in the computer program, which is included as successive pages of the appendix. We have subdivided the program by black lines and where the statements have a parallel in the transformational theory, that parallel is stated to the right on the print-out page. Otherwise the general function of the particular sequence of code is briefly stated.

The Basic Phonological Matrix

The program begins with the declaration statements, 1 through 5. These statements reserve space in the memory of the computer of the stated size for the variable names listed. For example, statement 2 reserves space for a two-dimensional array, to be addressed as MATRIX, of fifteen rows and thirty-six columns, with each intersection location occupied by a single character. FEATURE is likewise a two-dimensional array, CARD is a one-dimensional array, and I, J, K, L, M, and N are single variable locations filled by integers rather than character strings.

Statements 6-11 read in the distinctive feature matrix, called MATRIX, one column at a time. The top row of MATRIX, then, represents the underlying phonological segments of Russian. It should be noted at this point that we must employ a slightly specialized system of phonetic and phonemic symbolization, due to the restricted number of characters available on the key punch. Most symbols are the usual ones: /j/ represents the glide, /c/ the dental affricate sometimes written /ts/. For /č/ we use /?/, for /š/ the symbol /$/, and for /ž/ we use /H/. For the purely phonetic symbols we use [/] for [I], [Y] for [ɨ], [<] for [ə] , [@] for [ʌ] , and [%] for [ɛ].

Statements 6-11 fill MATRIX through column 25, representing the underlying phonological segments, or morphophonemes of the language.

Statements 12-20 print out MATRIX as defined above through column 25 with the feature labels spelled out at left. This output appears at the top of page 39 of the appendix. The labels are directed into the output in the proper format by the subroutine PRINT, which is invoked by statement 13, and of which the individual statements appear at the end of the program, as statements 461-532. This subroutine is invoked at each matrix print-out throughout the algorithm.

The Blank-filling Rules

Statements 21-108 encompass the blank-filling or context-free redundancy rules. Statements 21-75 apply to consonants and statements 76-108 apply to vowels. The machine is instructed to begin with column 1 of MATRIX and execute the computation of any statements whose conditions are met by the contents of that column.

Statement 22 instructs the machine to skip to statement 76 if the column under consideration is ⟨+vocalic⟩, that is, a true vowel or a liquid.

Statements 24 and 25 state that all consonants in Russian are non-low and non-round.

Statements 26-29 state that all anterior consonants (t, d, n, c, s, z, p, b, m, f, v) are non-back and non-high. By statements 30-36 anterior non-coronal non-strident consonants (p, b, m) are declared non-continuant.

The block of 37 and 38 declares anterior non-coronal strident consonants (f, v) to be continuant. This sequence illustrates the logic necessary to write an efficient algorithm. By the nature of statement 26, any segment processed between statement 26 and statement 39 (B2) must carry the feature ⟨+anterior⟩. Likewise any segment processed between statement 30 and statement 39 must be ⟨-coronal⟩. Statement 37 is the alternative or 'otherwise' to statement 32, which requires the segment to be ⟨-strident⟩. The only way statement 37 could be reached is if the condition of 32 proves false, and the segment is ⟨+strident⟩.

Statements 26-38 process the anterior consonants. If a segment is processed by this block, the machine jumps to statement 52 (B3) to continue processing. If the segment under consideration is non-anterior, processing would skip statements 28-38 and would continue with the block 39-51 which applies to non-anterior consonants.

Statement 39 declares all non-anterior consonants (č, š, ž, k, g, x) to be high. Statements 40-45 declare non-anterior coronal consonants (č, š, ž) to be strident and non-back. Statements 46-51 declare non-anterior non-coronal consonants (k, g, x) to be non-strident and back, and the non-anterior non-coronal continuant (x) to be voiceless.

Statement 52 declares coronal non-strident consonants (t, d, n) to be non-continuant. Statements 54-56 declare all strident consonants to be non-nasal.

Statement 57, after 53, declares strident non-continuant consonants (c, č) to be voiceless. Statement 59 declares back consonants (k, g, x) non-nasal.

Statements 61-72 specify the glide /j/ as non-anterior, non-coronal, high, non-back, non-strident, non-nasal, non-continuant, voiced, non-round and sharp. Statements 73-75 declare nasal consonants (n, m) to be voiced.

Statement 76 begins the blank-filling rules which apply to vocalic segments (including the liquids l, r). Statements 76-79 declare all ⟨+vocalic⟩ segments to be non-nasal, continuant and voiced. Statements 80-88 declare the liquids (l, r) to be coronal, non-high, non-low, non-back and non-round.

Statements 89-93 declare all true vowels to be non-anterior and non-coronal. Statements 94-99 declare low vowels (a) to be back and non-round.

Statement 100 declares high vowels (i, u) to be non-low. Statements 102-108 declare non-back vowels (e, i) as non-round and back vowels (o, u) as round. /a/ is not included in this last statement since statement 98 forced processing to jump to the end of the blank-filling rules, which are completed with statement 108.

Statements 109-114 read in the rest of MATRIX, filling columns 26 through 36 of the space alloted in the memory of the computer. These columns do not list morphophonemic segments, but rather phonetic segments which we will need for vowels in out final phonetic output (I, ɨ, ə, ʌ, 3); boundary symbols (# for end of text, a blank for word boundary, = for boundary after prepositions); a zero segment (Ø) (as intermediary to some glides--eliminated in final output; the asterisk (*) placed before a vowel to mark stress; and the comma (,) transliterated from the Russian soft sign, showing palatalization of the preceding segment. For some of these last columns, which specify phonetic detail at the lowest level, we use numerical values to specify a more precise degree of variation from the neutral position without the use of additional features, as suggested by Harms.[8]

Statements 113-124 print out this completed MATRIX with feature labels as found at the bottom of page 39 in the appendix.

[8] Harms, p. 15.

The Input

Statements 125-127 read in a portion of the text to be processed by the algorithm. The preparation of this text is relatively simple, and can be done by any key punch operator trained in Cyrillic-Latin transliteration with a few minimum instructions. The original text must be marked for stress and the = sign must be placed between each preposition and the following word. Otherwise, since Russian orthography is essentially morphophonemic a simple transliteration is all that is required. The system of transliteration to be followed is given below:

Cyrillic	Key Punch	Cyrillic	Key Punch
Й	J	Г	G
Т	T	Х	X
Д	D	Е	E (JE initially)
Н	N	Э	E
Ц	C	О	O
С	S	Ё	JO
З	Z	А	A
П	P	Я	JA
Б	B	И	I (JI initially)
М	M	Ы	I
Ф	F	У	U
В	V	Ю	JU
Ч	?	Ъ	omit
Ш	$	Ь	,

Cyrillic	Key Punch	Cyrillic	Key Punch
Щ	$?	Л	L
Ж	H	Р	R
К	K		

For the practical purpose of automatic phonetic transcription of a text, then, the * must be inserted at the key punch immediately before every stressed vowel (or before the glide J if it precedes the actual vowel), and the boundary = must replace the blank after each preposition. Taking this algorithm as the phonological component of a transformational-generative grammar, we would assume that the above information (stress, boundary symbols) was already present in the output of the morphological component.

The next step involves the input format. We want the machine to process one card of text at a time from input to output, so that we may leave the program unspecified as to the length of the text, instructing it to repeat, reading an indefinite number of cards until the text is finished. The text itself is punched continuously on the cards from column 1 through 80 of each card, ignoring card boundaries, with the exception that the first card of the text begins in column two. This allows for ease of key punching but causes a problem in computation. When the machine considers the final segment of the card under the context-sensitive rules it must know what the environment following that segment is. Therefore in statements 128-138 of our program we place a limit on the portion to be processed of the card already read in. Beginning in column 80 the machine reads backwards along the card. If it encounters the symbol '#' marking the end of the text, then we set the variable M equal to the column number in which the # appears. Otherwise the machine reads backward from the end of the card, setting M equal to the column number of the second blank encountered. We will ask the machine to subsequently process only the portion of the card from column 1 through column M. But since the remainder of the card has been read into the memory, the machine can examine the contents of columns near column M. After this part of the text has been completely processed, the array CARD is redefined for the next cycle. The portion of the actual card from column M to column 80 is moved to the beginning of the array CARD. The remainder of the array CARD is filled with non-introduced text from the next actual data card. The shortening operation occurs again, and processing continues.

Statements 143-151 copy CARD into the first row of FEATURE and fill the remainder of each column with the appropriate feature specifications copied from MATRIX after comparison of the head of the column of each matrix.

Statements 152-160 print out FEATURE as it is thus far defined, with feature labels, as found on top of page 40 in the appendix.

The Context-sensitive Rules

Now the context-sensitive rules apply to FEATURE, one segment (or column) at a time. Upon considering the transformational-generative theoretical model, we realize that each rule should be given a chance to apply to all segments before the following rule can apply. In other words, since the machine must consider one segment at a time we should run a DO-loop through all segments for each context-sensitive rule. The reason for this is that the context involved when considering any segment must be defined or processed for all previously applied rules before the current rule can apply to that particular segment. However, it happens that the context-sensitive rules for consonants involve only the environment following the consonant under consideration. Therefore we can save considerable program complexity and machine computation time by processing just consonants first, applying all the rules to a single segment before moving to the next segment but moving from right to left across the text (FEATURE). In this way the following environment will be completely processed as each consonant is considered.

The context-sensitive rules for vowels rely primarily although not exclusively, on the preceding environment, and if the consonants are processed first so that much of the environment is processed, we can logically process the vowels moving back from left to right across the text. (FEATURE).

Statement 165 instructs the machine to consider each segment from right to left, and statement 166 provides that if a segment is non-consonantal the machine goes on to the next segment.

Statements 168-182 specify /r/, /l/ and all anterior consonants other than /c/ as sharped before /i/ or /e/. Statement 183 instructs the machine to skip any further liquids ($\begin{bmatrix}\text{+consonantal}\\ \text{-vocalic}\end{bmatrix}$).

Statements 185-215 adjust voicing in obstruent clusters, where an obstruent is any non-nasal consonant. Statements 185-194 find the index of the first non-obstruent following the obstruent under consideration. Statements 195-201 provide that, if that first non-obstruent is a blank then all obstruents in the cluster are voiceless. Statements 202-208 provide that, if that first non-obstruent is the = boundary, then all obstruents in the cluster copy the voicing of the first segment following the = boundary. Statements 209-215 provide that, when the first non-obstruent is a sonorant (vowel, glide, liquid or nasal consonant) all obstruents in the cluster copy the voicing of the last obstruent in the cluster.

Statements 216-222 change anterior coronal strident consonants (c, s, z) to non-anterior (č, š, ž) before non-anterior consonants.

Statements 224-230 change the same anterior coronal strident consonants to non-anterior before the = boundary.

Statements 232-238 change /š/ to ⟨+sharp⟩ before /č/, and /ž/ to ⟨+sharp⟩ before /ž/.

Statements 240-245 sharp non-coronal consonants (p, b, m, f, v, k, g, x) before non-anterior coronal segments (č, š, ž).

Statements 247-258 sharp coronal non-strident consonants (t, d, n) before non-anterior coronal sharp segments and also sharp the coronal non-strident nasal consonant (n) before /š/ or /ž/.

Statements 260-266 specify coronal consonants other than /n/ (t, d, c, s, z, č, š, ž) as non-sharp before non-anterior coronal continuant consonants (š, ž).

Statements 268-271 declare anterior consonants as non-sharp before anterior coronal non-sharp consonants, and statements 273-276 declare anterior consonants non-sharp before non-sharp liquids.

Statements 278-284 sharp non-anterior non-coronal consonants (k, g, x) before non-low non-back non-consonantal segments (e, i, j).

Statements 286-292 declare anterior coronal consonants (t, d, n, c, s, z) sharped before sharped anterior consonantal segments.

Statements 294-296 declare non-coronal segments (p, b, m, f, v, k, g, x) as non-sharp before /r/ and any non-sharped segments.

Statements 298-301 sharp /x/ before sharped non-anterior non-coronal segments. Statement 303 declares non-coronal consonants as non-sharp before /l/.

Statements 305-308 sharp non-coronal consonants before sharped coronal segments. Statements 310-313 sharp a consonant before its sharped geminate.

Statements 315-324 change anterior coronal non-strident consonants (t, d, n) to zero [ø] between anterior coronal continuant consonants (s,z) and /n/. This environment overlapping is possible when moving from right to left, since it is executed as the machine has the left-most segment (anterior coronal continuant) under consideration.

Statements 326-334 declare non-anterior non-coronal consonants (k, g, x) non-sharp before segments which are not non-low non-back and non-consonantal (e, i, j).

Next follow the context-sensitive rules for ⟨+vocalic⟩ segments. Statements 335-360 apply to stressed vowels. Statements 339-347 change stressed /i/ to ⟨+back⟩ ([ɨ]) following a non-sharp consonant, and statements 348-356 change stressed /e/ to ⟨+low⟩ ([ɛ]) preceding a non-sharp consonant.

Statements 361 & 363 restrict the subsequent rules to true vowels ($\begin{bmatrix}+\text{vocalic}\\-\text{consonantal}\end{bmatrix}$). The remainder of the context-sensitive rules for vowels apply to unstressed vowels.

Statements 365-372 change non-high non-low vowels (e, o) to high and non-rounded ([i]) following high coronal segments (č, š, ž).

Statements 373-378 change all non-high non-low vowels (e, o) to low back non-rounded ([a]).

Statements 379-404 process the non-high low vowel (/a/). Statements 381-386 determine whether the vowel occurs pretonically, initially, or in some other position. Statements 387-391 change /a/ to ⟨2low⟩ ([ʌ]) pretonically, and statements 392-396 do the same for /a/ in word-initial position. Statements 397-404 change /a/ to high non-low non-back (/i/) after sharped segments and to non-low ([ə]) after non-sharp segments (when not pretonic or word-initial).

Statements 405-512 finally change the high non-low non-back vowel (/i/) to ⟨2back⟩ ([I]) after sharped segments. This ends the context-sensitive rules for Russian, and ends the basic computation section of the algorithm.

Output

Statements 413-421 next replace the first row of FEATURE with segments from the first row of MATRIX by comparing the other rows of each column of FEATURE with each column of MATRIX.

Statements 422-449 now print FEATURE, the phonetic transcription of the Russian text showing all feature specifications. Feature labels are added at the left by invoking the subroutine PRINT, the symbol ',' is added to the first row after each sharped segment, and Ø segments are omitted in the output.

If the last column of FEATURE contains the #, then the program ends, as directed by statement 450. Otherwise, the array CARD is redefined as described above, more text is read in, and the context-sensitive rules are applied to the new text.

The program as described, then, represents an algorithm for the automatic phonetic transcription of a Russian text, and as such is a working model of the phonological component of a transformational-generative grammar of Russian.

THE FOLLOWING DOCUMENT(S) IS OVERSIZED AND IS BEING FILMED IN SECTIONS TO INSURE COMPLETENESS AND CONTINUITY

```
PHONOL: PROC OPTIONS(MAIN);

          PHONOL: PROC OPTIONS(MAIN);
          DCL MATRIX(15,36) CHAR(1);
          DCL FEATURE(15,80) CHAR(1);
          DCL CARD(80) CHAR(1);
          DCL (I,J,K,L,M,N) FIXED BIN;

          DO J=1 TO 25;
             DO I=1 TO 15;
             GET EDIT(MATRIX(I,J))(A(1));
             END;
             GET SKIP;
          END;
          DO I=1 TO 15;
          CALL PRINT;
          A: PUT EDIT(MATRIX(I,1))(COLUMN(20),A(1));
             DO J=2 TO 25;
             PUT EDIT(MATRIX(I,J))(X(1),A(1));
             END;
             PUT SKIP;
          END;
          PUT SKIP;

          DO J=1 TO 25;
          IF MATRIX(2,J)¬='-' THEN GO TO D;
          MATRIX(7,J)='-';
          MATRIX(13,J)='-';
          IF MATRIX(4,J)¬='+' THEN GO TO B2;
          MATRIX(8,J)='-';
          MATRIX(6,J)='-';
          IF MATRIX(5,J)¬='-' THEN GO TO B3;
          IF MATRIX(9,J)='-' THEN
             DO;
                 MATRIX(11,J)='-';
                 GO TO B3;
             END;
          MATRIX(11,J)='+';
          GO TO B3;
          B2: MATRIX(6,J)='+';
          IF MATRIX(5,J)='+' THEN
             DO;
                 MATRIX(8,J)='-';
                 MATRIX(9,J)='+';
                 GO TO B3;
             END;
             ELSE DO;
             MATRIX(8,J)='+';
             MATRIX(9,J)='-';
             IF MATRIX(11,J)='+' THEN MATRIX(12,J)='-';
             END;
          B3: IF MATRIX(5,J)='+' & MATRIX(9,J)='-' THEN MATRIX(1
          IF MATRIX(9,J)¬='+' THEN GO TO B4;
          MATRIX(10,J)='-';
          IF MATRIX(11,J)='-' THEN MATRIX(12,J)='-';
          B4: IF MATRIX(8,J)='+' THEN MATRIX(10,J)='-';
          IF MATRIX(3,J)='-' THEN
             DO;
                 MATRIX(4,J)='-';
```

Declarations

Read in MATRIX (morphophoneme matrix)

`;`

Print MATRIX with feature labels

Blank-filling rules through statement 108
Jump to D if +vocalic — all before D are —vocalic

$\begin{bmatrix}-\text{vocalic}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{low}\\-\text{round}\end{bmatrix}$

$\begin{bmatrix}+\text{anterior}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{bock}\\-\text{high}\end{bmatrix}$

$\begin{bmatrix}+\text{anterior}\\-\text{coronal}\\-\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{continuant}\end{bmatrix}$

$\begin{bmatrix}+\text{anterior}\\-\text{coronal}\\+\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{continuant}\end{bmatrix}$

$\begin{bmatrix}-\text{anterior}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{high}\end{bmatrix}$

$\begin{bmatrix}-\text{anterior}\\+\text{coronal}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{back}\\+\text{strident}\end{bmatrix}$

$\begin{bmatrix}-\text{anterior}\\-\text{coronal}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{back}\\-\text{strident}\end{bmatrix}$

`'-';`

`THEN MATRIX(I,J)='-';`

$\begin{bmatrix}+\text{coronal}\\-\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{continuant}\end{bmatrix}$

$\begin{bmatrix}+\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{nasal}\end{bmatrix}$

`;`

$\begin{bmatrix}+\text{strident}\\-\text{continuant}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{voiced}\end{bmatrix}$

`'-';`

$\begin{bmatrix}+\text{bock}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{nosal}\end{bmatrix}$

```
             MATRIX(5,J)='-';
             MATRIX(6,J)='+';
             MATRIX(8,J)='-';
             MATRIX(9,J)='-';
             MATRIX(10,J)='-';
             MATRIX(11,J)='-';
             MATRIX(12,J)='+';
             MATRIX(13,J)='-';
        END;
      IF MATRIX(10,J)='+' THEN MATRIX(12,J)='+';
      GO TO F;
      D: MATRIX(9,J)='-';
        MATRIX(10,J)='-';
        MATRIX(11,J)='+';
        MATRIX(12,J)='+';
      IF MATRIX(3,J)='+' THEN
        DO;
             MATRIX(5,J)='+';
             MATRIX(6,J)='-';
             MATRIX(7,J)='-';
             MATRIX(8,J)='-';
             MATRIX(13,J)='-';
             GO TO D2;
        END;
      IF MATRIX(3,J)='-' THEN
        DO;
      MATRIX(4,J)='-';
      MATRIX(5,J)='-';
        END;
      D2: IF MATRIX(7,J)='+' THEN
        DO;
             MATRIX(8,J)='+';
             MATRIX(13,J)='-';
             GO TO F;
        END;
      IF MATRIX(6,J)='+' THEN MATRIX(7,J)='-';
      IF MATRIX(8,J)='-' THEN
        DO;
             MATRIX(13,J)='-';
             GO TO F;
        END;
      MATRIX(13,J)='+';
      F: END;

      DO J=26 TO 36;
             DO I=1 TO 15;
             GET EDIT(MATRIX(I,J))(A(1));
             END;
             GET SKIP;
        END;
      DO I=1 TO 15;
      CALL PRINT;
      PUT EDIT(MATRIX(I,1))(COLUMN(20),A(1));
        DO J=2 TO 36;
        PUT EDIT(MATRIX(I,J))(X(1),A(1));
        END;
        PUT SKIP;
```

$$\begin{bmatrix} -\text{consonantal} \end{bmatrix} \longrightarrow \begin{bmatrix} -\text{anterior} \\ -\text{coronal} \\ +\text{high} \\ -\text{back} \\ -\text{strident} \\ -\text{nasal} \\ -\text{continuant} \end{bmatrix}$$

$$\begin{bmatrix} +\text{nasal} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{voiced} \end{bmatrix}$$

$$\begin{bmatrix} +\text{vocalic} \end{bmatrix} \longrightarrow \begin{bmatrix} -\text{nasal} \\ +\text{continuant} \\ +\text{voiced} \end{bmatrix}$$

$$\begin{bmatrix} +\text{vocalic} \\ +\text{consonantal} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{coronal} \\ -\text{high} \\ -\text{low} \\ -\text{back} \\ -\text{round} \end{bmatrix}$$

$$\begin{bmatrix} +\text{vocalic} \\ -\text{consontal} \end{bmatrix} \longrightarrow \begin{bmatrix} -\text{anterior} \\ -\text{coronal} \end{bmatrix}$$

$$\begin{bmatrix} +\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{back} \\ -\text{round} \end{bmatrix}$$

$$\begin{bmatrix} +\text{high} \end{bmatrix} \longrightarrow \begin{bmatrix} -\text{low} \end{bmatrix}$$

$$\begin{bmatrix} \alpha\ \text{back} \end{bmatrix} \longrightarrow \begin{bmatrix} \alpha\ \text{round} \end{bmatrix}$$

Read in remainder of MATRIX for more complete phonetic capability

Print complete MATRIX with feature labels

```
END;
PUT SKIP;
PUT PAGE;

G: DO I=1 TO 80;
GET EDIT(CARD(I))(A(1));
END;
G1: L=0;
CO M=80 TO 1 BY -1;
IF CARD(M)='#' THEN GO TO H;
END;
CO M=80 TO 1 BY -1;
IF CARD(M)=' ' THEN L=L+1;
IF L=2 THEN GO TO H;
END;
H: DO I=1 TO M;
   PUT EDIT(CARD(I))(A(1));
   END;
PUT SKIP;
CO I=1 TO M;
   DO J=1 TO 36;
   IF CARD(I)=MATRIX(1,J) THEN
       DO;
       FEATURE(*,I)=MATRIX(*,J);
       GO TO P;
       END;
   END;
P: END;

CO I=1 TO 15;
CALL PRINT;
PUT EDIT(FEATURE(I,1))(COLUMN(20),A(1));
   DO J=2 TO M;
   PUT EDIT(FEATURE(I,J))(X(1),A(1));
   END;
   PUT SKIP;
END;
PUT SKIP;

CO J=1 TO M-1;
IF FEATURE(1,J+1)=',' THEN FEATURE(14,J)='+';
END;

CONS: DO J=(M-1) TO 1 BY -1;
IF FEATURE(3,J)¬='+' THEN GO TO C10;
IF FEATURE(2,J)='+' THEN GO TC C1;
IF FEATURE(4,J)='+' & FEATURE(1,J)¬='C' THEN GO TO C1;
ELSE GO TO C2;
C1: IF FEATURE(1,J+1)='E' | FEATURE(1,J+1)='I' THEN GO T
IF FEATURE(1,J+1)='*' & FEATURE(1,J+2)='I' THEN GO TO C1
IF FEATURE(1,J+1)='*' & FEATURE(1,J+2)='E' THEN GO TO C1
IF FEATURE(1,J+1)='J' THEN GO TO C101;
ELSE GO TO C2;
C101: FEATURE(14,J)='+';
C2: IF FEATURE(2,J)¬='-' THEN GO TO C10;
IF FEATURE(10,J)='-' THEN GO TO C3;
ELSE GO TO Y;
```

Read in CARD

Shorten CARD to stop at '#'
or at second blank from end

Print CARD as defined above

CARD becomes first row of FEATURE matrix

Copy specifications from MATRIX into appropriate columns of FEATURE

Print FEATURE with feature labels

$$[+\text{seg}] \rightarrow [+\text{sharp}] / \underline{\quad\quad} ,$$

Context-sensitive Rules for consonants
if segment not ⟨+consonantal⟩ jump to 334 (next segment)

```
O TO C1;

 THEN GO TO C101;
N GO TO C101;
N GO TO C101;
```

$$\left\{\begin{matrix} R \\ L \\ \langle +\text{anterior} \rangle \\ \text{not}/c/ \end{matrix}\right\} \rightarrow [+\text{sharp}] / \underline{\quad\quad}^{(*)} \left\{\begin{matrix} I \\ E \end{matrix}\right\}$$

if segment a liquid jump to 334 (next segment

```
   C3: DO K=(J+1) TO 80;
      IF FEATURE(2,K)='-' & FEATURE(3,K)='+' THEN GO TO C4;
   ELSE GO TO Q1;
   C4: IF FEATURE(10,K)='-' THEN GO TO Y1;
   ELSE GO TO Q2;
   C1: IF FEATURE(1,K)=' ' THEN
      DO;
      DO N=(K-1) TO J BY -1;
      FEATURE(12,N)='-';
      GO TO Y;
      END;
      END;
   IF FEATURE(1,K)='=' THEN
      DO;
      DO N=(K-1) TO J BY -1;
          FEATURE(12,N)=FEATURE(12,K+1);
      GO TO Y;
      END;
      END;
      Q2: IF K=J+1 THEN GC TO Y;
      DO N=(K-2) TO J BY -1;
          FEATURE(12,N)=FEATURE(12,K-1);
      GO TO Y;
      END;
   Y1: END;
   Y: IF FEATURE(5,J)='+' & FEATURE(4,J)='+' THEN GO TO C5;
   ELSE GO TO C6;
   C5: IF FEATURE(9,J)='+' & FEATURE(3,J+1)='+' THEN GO TO
   ELSE GO TO C6;
   C7: IF FEATURE(4,J+1)='-' & FEATURE(5,J+1)='+' THEN FEAT
   C6: IF FEATURE(5,J)='+' & FEATURE(4,J)='+' THEN GO TO C8
   ELSE GO TO C9;
   C8: IF FEATURE(9,J)='+' & FEATURE(1,J+1)='=' THEN GO TO
   ELSE GO TO C9;
   C10: IF FEATURE(5,J+2)='+' & FEATURE(4,J+2)='-' THEN FEA
   C9: IF FEATURE(1,J)='$' & FEATURE(1,J+1)='?' THEN GO TO
   IF FEATURE(1,J)¬='H' | FEATURE(1,J+1)¬='H' THEN GO TO Q6
   Q3: FEATURE(14,J)='+';
   FEATURE(14,J+1)='+';
   IF FEATURE(1,J+1)='?' THEN FEATURE(11,J+1)='+';
   Q6: IF FEATURE(5,J)¬='+' THEN GO TO Q7;
   IF FEATURE(5,J+1)='+' & FEATURE(4,J+1)='-' THEN GO TO C1
   ELSE GO TO Q7;
   C11: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)='+';
   Q7: IF FEATURE(9,J)='-' & FEATURE(5,J)='+' THEN GO TO Q8;
   GO TO Q9;
   Q8: IF FEATURE(5,J+1)='+' & FEATURE(4,J+1)='-' THEN GO T
   ELSE GO TO C13;
   C12: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)='+';
   C13: IF FEATURE(10,J)='+' & FEATURE(1,J+1)='$' THEN FEAT
   GO TO Q9;
   IF FEATURE(10,J)='+' & FEATURE(1,J+1)='H' THEN FEATURE(1
   Q9: IF FEATURE(5,J)='+' & FEATURE(1,J)¬='N' THEN GO TO C
   ELSE GO TO C15;
   C14: IF FEATURE(11,J+1)='+' & FEATURE(5,J+1)='+' THEN GO
   ELSE GO TO C15;
   C16: IF FEATURE(4,J+1)='-' THEN FEATURE(14,J)='-';
```

```
GO TO C4;
```

if segment an obstruent (non-nasal consonant) find first non-obstruent following

$$\begin{bmatrix}-\text{nasal}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{voiced}\end{bmatrix} \Big/ \_\_\_ \not{b}$$

$$\begin{bmatrix}-\text{nasal}\end{bmatrix} \rightarrow \begin{bmatrix}\alpha\ \text{voiced}\end{bmatrix} \Big/ \_\_\_ = \begin{bmatrix}+\text{seg} \\ \alpha\ \text{voiced}\end{bmatrix}$$

$$\begin{bmatrix}-\text{nasal}\end{bmatrix} \rightarrow \begin{bmatrix}\alpha\ \text{voiced}\end{bmatrix} \Big/ \_\_\_ \begin{bmatrix}+\text{cons.} \\ -\text{nasal} \\ \alpha\ \text{voiced}\end{bmatrix} \begin{bmatrix}-\text{obstruent}\end{bmatrix}$$

```
GO TO C5;

EN GO TO C7;

THEN FEATURE(4,J)='-';
```

$$\begin{bmatrix}+\text{anterior} \\ +\text{coronal} \\ +\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{anterior}\end{bmatrix} \Big/ \_\_\_ \begin{bmatrix}C \\ -\text{anterior}\end{bmatrix}$$

```
 GO TO C8;

EN GO TO C10;

 THEN FEATURE(4,J)='-';
```

$$\begin{bmatrix}+\text{anterior} \\ +\text{coronal} \\ +\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}-\text{anterior}\end{bmatrix} \Big/ \_\_\_ =$$

```
EN GO TO C13;
 GO TO C6;
```

$$\begin{Bmatrix}\check{s} \\ \check{z}\end{Bmatrix} \rightarrow \begin{bmatrix}+\text{sharp}\end{bmatrix} \Big/ \_\_\_ \begin{Bmatrix}\check{c} \\ \check{z}\end{Bmatrix}$$

```
 GO TO C11;

'+';
```

$$\begin{bmatrix}-\text{coronal}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{sharp}\end{bmatrix} \Big/ \_\_\_ \begin{bmatrix}-\text{anterior} \\ +\text{coronal}\end{bmatrix}$$

```
 GO TO C8;

THEN GO TO C12;
```

$$\begin{bmatrix}+\text{coronal} \\ -\text{strident}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{sharp}\end{bmatrix} \Big/ \_\_\_ \begin{bmatrix}-\text{anterior} \\ +\text{coronal} \\ +\text{sharp}\end{bmatrix}$$

```
'+';
THEN FEATURE(14,J)='+';

FEATURE(14,J)='+';
```

$$\begin{bmatrix}+\text{coronal} \\ -\text{strident} \\ +\text{nasal}\end{bmatrix} \rightarrow \begin{bmatrix}+\text{sharp}\end{bmatrix} \Big/ \_\_\_ \begin{Bmatrix}\check{s} \\ \check{z}\end{Bmatrix}$$

```
N GO TO C14;

' THEN GO TO C16;

-';
```

$$\begin{bmatrix}+\text{coronal} \\ \text{not } N\end{bmatrix} \rightarrow \begin{bmatrix}-\text{sharp}\end{bmatrix} \Big/ \_\_\_ \begin{bmatrix}-\text{anterior} \\ +\text{coronal} \\ +\text{continuant}\end{bmatrix}$$

```
PHONOL: PRCC OPTIONS(MAIN);

          C15: IF FEATURE(4,J)='+' & FEATURE(4,J+1)='+' THEN GO TO
          ELSE GO TO C18;
          U: IF FEATURE(5,J+1)='+' & FEATURE(14,J+1)='-' THEN FEATU
          C18: IF FEATURE(4,J)='+' & FEATURE(2,J+1)='+' THEN GO TO
          ELSE GO TO C20;
          W: IF FEATURE(3,J+1)='+' & FEATURE(14,J+1)='-' THEN FEAT
          C20: IF FEATURE(4,J)='-' & FEATURE(5,J)='-' THEN GO TO C
          ELSE GO TO C22;
          C21: IF FEATURE(3,J+1)='-' & FEATURE(7,J+1)='-' THEN GO
          ELSE GO TO C22;
          C23: IF FEATURE(8,J+1)='-' THEN FEATURE(14,J)='+';
          C22: IF FEATURE(4,J)='+' & FEATURE(5,J)='+' THEN GO TO Z;
          ELSE GO TO C27;
          Z: IF FEATURE(3,J+1)='+' & FEATURE(4,J+1)='+' THEN GO TO
          ELSE GO TO C27;
          C25: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)='+';
          C27: IF FEATURE(5,J)='-' & FEATURE(14,J+1)='+' THEN FEATU
          IF FEATURE(5,J)='-' & FEATURE(1,J+1)='R' THEN FEATURE(14.
          IF FEATURE(1,J)='X' & FEATURE(5,J+1)='-' THEN GO TO X;
          ELSE GO TO C29;
          X: IF FEATURE(4,J+1)='-' & FEATURE(14,J+1)='+' THEN FEATU
          C29: IF FEATURE(5,J)='-' & FEATURE(1,J+1)='L' THEN FEATU-
          IF FEATURE(5,J)='-' & FEATURE(5,J+1)='+' THEN GO TO C30;
          ELSE GO TO C31;
          C30: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)='+';
          C31: IF FEATURE(1,J)=FEATURE(1,J+1) THEN GO TO C32;
          ELSE GO TU C33;
          C32: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)='+';
          C33: IF FEATURE(4,J)='+' & FEATURE(5,J)='+' THEN GO TO C
          ELSE GO TO C35;
          C34: IF FEATURE(11,J)='+' & FEATURE(5,J+1)='+' THEN GO TO
          ELSE GO TO C35;
          C341: IF FEATURE(4,J+1)='+' & FEATURE(9,J+1)='-' THEN GO
          ELSE GO TO C35;
          C342: IF FEATURE(1,J+2)='N' THEN FEATURE(1,J+1)='O';
          C35: IF FEATURE(4,J)='-' & FEATURE(5,J)='-' THEN GO TO C
          ELSE GO TO Q10;
          C36: IF FEATURE(3,J+1)¬='-' & FEATURE(7,J+1)¬='-' THEN GO
          ELSE GO TO Q10;
          C37: IF FEATURE(8,J+1)¬='-' THEN FEATURE(14,J)='-';
          Q10: END CONS;

          VOW: DO J=1 TO (M-1);
          IF FEATURE(1,J)='*' THEN GO TC V1;
          ELSE GO TO V41;
          V1: IF FEATURE(1,J+1)='I' THEN
             DO;
             IF FEATURE(3,J-1)='+' & FEATURE(14,J-1)='-' THEN
             DO;
             FEATURE(8,J+1)='+';
             J=J+1;
             GO TO Q13;
             END;
             END;
          V2: IF FEATURE(1,J+1)='E' THEN
             DO;
          IF FEATURE(3,J+2)='+' & FEATURE(14,J+2)='-' THEN
```

```
PHONOL: PROC OPTIONS(MAIN);

            C15: IF FEATURE(4,J)='+' & FEATURE(4,J+1)='+'
            ELSE GO TO C18;
            U: IF FEATURE(5,J+1)='+' & FEATURE(14,J+1)='-'
            C18: IF FEATURE(4,J)='+' & FEATURE(2,J+1)='+'
            ELSE GO TO C20;
            W: IF FEATURE(3,J+1)='+' & FEATURE(14,J+1)='-'
            C20: IF FEATURE(4,J)='-' & FEATURE(5,J)='-' TH
            ELSE GO TO C22;
            C21: IF FEATURE(3,J+1)='-' & FEATURE(7,J+1)='-
            ELSE GO TO C22;
            C23: IF FEATURE(8,J+1)='-' THEN FEATURE(14,J)=
            C22: IF FEATURE(4,J)='+' & FEATURE(5,J)='+' TH
            ELSE GO TO C27;
            Z: IF FEATURE(3,J+1)='+' & FEATURE(4,J+1)='+'
            ELSE GO TO C27;
            C25: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)
            C27: IF FEATURE(5,J)='-' & FEATURE(14,J+1)='+'
            IF FEATURE(5,J)='-' & FEATURE(1,J+1)='R' THEN
            IF FEATURE(1,J)='X' & FEATURE(5,J+1)='-' THEN
            ELSE GO TO C29;
            X: IF FEATURE(4,J+1)='-' & FEATURE(14,J+1)='+'
            C29: IF FEATURE(5,J)='-' & FEATURE(1,J+1)='L'
            IF FEATURE(5,J)='-' & FEATURE(5,J+1)='+' THEN
            ELSE GO TO C31;
            C30: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)
            C31: IF FEATURE(1,J)=FEATURE(1,J+1) THEN GO TO
            ELSE GO TU C33;
            C32: IF FEATURE(14,J+1)='+' THEN FEATURE(14,J)
            C33: IF FEATURE(4,J)='+' & FEATURE(5,J)='+' TH
            ELSE GO TO C35;
            C34: IF FEATURE(11,J)='+' & FEATURE(5,J+1)='+'
            ELSE GO TO C35;
            C341: IF FEATURE(4,J+1)='+' & FEATURE(9,J+1)='
            ELSE GO TO C35;
            C342: IF FEATURE(1,J+2)='N' THEN FEATURE(1,J+1
            C35: IF FEATURE(4,J)='-' & FEATURE(5,J)='-' TH
            ELSE GO TO Q10;
            C36: IF FEATURE(3,J+1)¬='-' & FEATURE(7,J+1)¬=
            ELSE GO TO Q10;
            C37: IF FEATURE(8,J+1)¬='-' THEN FEATURE(14,J)
            Q10: END CONS;

            VOW: DO J=1 TO (M-1);
            IF FEATURE(1,J)='*' THEN GO TO V1;
            ELSE GO TO V41;
            V1: IF FEATURE(1,J+1)='I' THEN
               DO;
               IF FEATURE(3,J-1)='+' & FEATURE(14,J-1)='-'
               DO;
               FEATURE(8,J+1)='+';
               J=J+1;
               GO TO Q13;
               END;
               END;
            V2: IF FEATURE(1,J+1)='E' THEN
               DO;
            IF FEATURE(3,J+2)='+' & FEATURE(14,J+2)='-' TH
```

```
   DO;
   FEATURE(7,J+1)='+';
   J=J+1;
   GO TO Q13;
   END;
   END;
V4: IF FEATURE(1,J+1)='J' THEN J=J+2;
ELSE J=J+1;
GO TO Q13;
V41: IF FEATURE(2,J)¬='+' THEN GO TO Q13;
IF FEATURE(3,J)¬='-' THEN GO TO Q13;
IF FEATURE(6,J)='-' & FEATURE(7,J)='-' THEN GO TO V5;
ELSE GO TO V6;
V5: IF FEATURE(5,J-1)='+' & FEATURE(6,J-1)='+' THEN
   DO;
   FEATURE(6,J)='+';
   FEATURE(13,J)='-';
   END;
V6: IF FEATURE(6,J)='-' & FEATURE(7,J)='-' THEN
   DO;
   FEATURE(7,J)='+';
   FEATURE(8,J)='+';
   FEATURE(13,J)='-';
   END;
IF FEATURE(6,J)='-' & FEATURE(7,J)='+' THEN
   DO;
   DO N=1 TO 7;
   IF FEATURE(1,J+N)=' ' THEN GO TO V9;
   IF FEATURE(2,J+N)='+' & FEATURE(3,J+N)='-' THEN GO TO
   END;
V7: IF FEATURE(1,(J+N)-1)='*' & FEATURE(14,J-1)='-' THEN
       DO;
       FEATURE(7,J)='2';
       GO TO V12;
       END;
   IF FEATURE(1,J-1)=' ' THEN
       DO;
       FEATURE(7,J)='2';
       GO TO V12;
       END;
V9: IF FEATURE(14,J-1)='+' THEN
   DO;
   FEATURE(6,J)='+';
   FEATURE(7,J)='-';
   FEATURE(8,J)='-';
   END;
ELSE FEATURE(7,J)='-';
   END;
V12: IF FEATURE(6,J)='+' & FEATURE(7,J)='-' THEN GO TO V
ELSE GO TO Q13;
V13: IF FEATURE(8,J)='-' & FEATURE(14,J-1)='+' THEN GO T
ELSE GO TO Q13;
V14: FEATURE(8,J)='2';
Q13: END VOW;

DO J=1 TO M;
   DO K=1 TO 36;
```

$$E \longrightarrow \begin{bmatrix} +\text{low} \end{bmatrix} \Big/ * \_\_\_ \begin{bmatrix} C \\ -\text{sharp} \end{bmatrix}$$

machine jumps to segment following vowel

if segment not <+ vocalic> get next segment

if segment not <-consonantal> get next segment

TO V5;

THEN

$$\begin{bmatrix} -\text{high} \\ -\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{high} \\ -\text{round} \end{bmatrix} \Big/ \_\_\_ \begin{bmatrix} +\text{coronal} \\ +\text{high} \end{bmatrix}$$

N

$$\begin{bmatrix} -\text{high} \\ -\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{low} \\ +\text{back} \\ -\text{round} \end{bmatrix}$$

THEN GO TO V7;

)='-' THEN

$$\begin{bmatrix} -\text{high} \\ +\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} 2\,\text{low} \end{bmatrix} \Big/ \_\_\_ \; C_0 * V$$

$$\begin{bmatrix} -\text{high} \\ +\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} 2\,\text{low} \end{bmatrix} \Big/ \not{b} \; \_\_\_$$

$$\begin{bmatrix} -\text{high} \\ +\text{low} \end{bmatrix} \longrightarrow \begin{bmatrix} +\text{high} \\ -\text{low} \\ -\text{back} \end{bmatrix} \Big/ \begin{bmatrix} +\text{sharp} \end{bmatrix} \_\_\_$$

$$\begin{bmatrix} -\text{low} \end{bmatrix} \Big/ \begin{bmatrix} -\text{sharp} \end{bmatrix} \_\_\_$$

IEN GO TO V13;

THEN GO TO V14;

$$\begin{bmatrix} +\text{high} \\ -\text{low} \\ -\text{back} \end{bmatrix} \longrightarrow \begin{bmatrix} 2\,\text{back} \end{bmatrix} \Big/ \begin{bmatrix} +\text{sharp} \end{bmatrix} \_\_\_$$

```
       DO I=2 TO 13;
       IF FEATURE(I,J)¬=MATRIX(I,K) THEN GO TO R;
       END;
       FEATURE(1,J)=MATRIX(1,K);
   R: END;
END;

DO I=1 TO 15;
CALL PRINT;
IF FEATURE(1,1)='O' THEN GO TO R1;
PUT EDIT(FEATURE(I,1))(COLUMN(20),A(1),A(1));
IF FEATURE(I,1)='#' THEN GO TO FIN;
IF FEATURE(14,1)='+' THEN
   DO;
   IF I=1 THEN PUT EDIT(',')(A(1));
   ELSE PUT EDIT(' ')(A(1));
   GO TO R1;
   END;
R1: DO J=2 TO M;
IF FEATURE(1,J)='O' THEN GO TO R2;
PUT EDIT(FEATURE(I,J))(X(1),A(1));
IF FEATURE(14,J)='+' & FEATURE(1,J)¬='J' & FEATURE(1,J)-
   DO;
   IF I=1 THEN PUT EDIT(',')(A(1));
   ELSE PUT EDIT(' ')(A(1));
   GO TO R2;
   END;
R2: END;
PUT SKIP;
END;
IF FEATURE(1,M)='#' THEN GO TO FIN;

CARD(1)=CARD(M);
DO I=1 TO (80-M);
CARD(I)=CARD(M+I);
END;
T: I=I+1;
GET EDIT (CARD(I))(A(1));
IF I=80 THEN GO TO G1;
GO TO T;

PRINT: PROCEDURE;
IF I=2 THEN
   DO;
       PUT EDIT('VOCALIC')(A(7));
       RETURN;
   END;
IF I=3 THEN
   DO;
       PUT EDIT('CONSONANTAL')(A(11));
       RETURN;
   END;
IF I=4 THEN
   DO;
       PUT EDIT('ANTERIOR')(A(8));
       RETURN;
   END;
```

```
R;
```

Replace top row of FEATURE with segments from top row of MATRIX by comparing other rows

Print FEATURE adding "," after sharped segments and omitting ∅ segments

```
TURE(1,J)¬=',' THEN
```

If last column of FEATURE is "#" - end program

Redefine CARD starting with ƀ from end of previous CARD, then segments left off previous CARD.

Fill out remainder of CARD from unprocessed input. Return to context-sensitive rules

Subroutine to print feature labels in appropriate format with each matrix print-out

```
IF I=5 THEN
   DO;
       PUT EDIT('CORONAL')(A(7));
       RETURN;
   END;
IF I=6 THEN
   DO;
       PUT EDIT('HIGH')(A(4));
       RETURN;
   END;
IF I=7 THEN
   DO;
       PUT EDIT('LOW')(A(3));
       RETURN;
   END;
IF I=8 THEN
   DO;
       PUT EDIT('BACK')(A(4));
       RETURN;
   END;
IF I=9 THEN
   DO;
       PUT EDIT('STRIDENT')(A(8));
       RETURN;
   END;
IF I=10 THEN
   DO;
       PUT EDIT('NASAL')(A(5));
       RETURN;
   END;
IF I=11 THEN
   DO;
       PUT EDIT('CONTINUANT')(A(10));
       RETURN;
   END;
IF I=12 THEN
   DO;
       PUT EDIT('VOICED')(A(6));
       RETURN;
   END;
IF I=13 THEN
   DO;
       PUT EDIT('ROUND')(A(5));
       RETURN;
   END;
IF I=14 THEN
   DO;
       PUT EDIT('SHARP')(A(5));
       RETURN;
   END;
IF I=15 THEN
   DO;
       PUT EDIT('STRESSED')(A(8));
       RETURN;
   END;
END PRINT;
```

```
         PHONOL: PRCC OPTIONS(MAIN);

  533              FIN: END PHONOL;
```

End of algorithm

MATRIX

	J	T	D	N	C	S	Z	P	B	M	F	V	?	$	H	K	G	X	E	O	A	I	U	R	L
VOCALIC	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+
CONSONANTAL	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	+	+
ANTERIOR	0	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	−	0	0	0	0	0	−	+
CORONAL	0	+	+	+	+	+	+	−	−	−	−	−	+	+	+	−	−	−	0	0	0	0	0	0	0
HIGH	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	−	−	+	+	0	0
LOW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	−	+	0	0	0	0
BACK	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	−	+	0	−	+	0	0
STRIDENT	0	−	−	−	+	+	+	−	−	−	+	+	0	0	0	0	0	0	0	0	0	0	0	0	0
NASAL	0	−	−	+	0	0	0	−	−	+	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
CONTINUANT	0	0	0	0	−	+	+	0	0	0	0	0	−	+	+	−	−	+	0	0	0	0	0	0	0
VOICED	0	−	+	0	0	−	+	−	+	0	−	+	0	−	+	−	+	0	0	0	0	0	0	0	0
ROUND	0	0	−	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
SHARP	+				−								+	−	−										
STRESSED																			+	+		+			

	J	T	D	N	C	S	Z	P	B	M	F	V	?	$	H	K	G	X	E	O	A	I	U	R	L	/	Y	<	ə	:
VOCALIC	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	+	+	+	+	+	-
CONSONANTAL	−	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	+	+	−	−	−	−	-
ANTERIOR	−	+	+	+	+	+	+	+	+	+	+	+	−	−	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	-
CORONAL	−	+	+	+	+	+	+	−	−	−	−	−	+	+	+	−	−	−	−	−	−	−	−	+	+	−	−	−	−	-
HIGH	+	−	−	−	−	−	−	−	−	−	−	−	+	+	+	+	+	+	−	−	−	+	+	−	−	+	+	−	−	-
LOW	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	−	−	−	−	−	−	−	2	+
BACK	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	+	+	−	+	+	−	+	−	−	2	+	+	+	-
STRIDENT	−	−	−	−	+	+	+	−	−	−	+	+	+	+	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	-
NASAL	−	−	−	+	−	−	−	−	−	+	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	-
CONTINUANT	−	−	−	−	−	+	+	−	−	−	+	+	−	+	+	−	−	+	+	+	+	+	+	+	+	+	+	+	+	+
VOICED	+	−	+	+	−	−	+	−	+	+	−	+	−	−	+	−	+	−	+	+	+	+	+	+	+	+	+	+	+	+
ROUND	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	−	+	−	−	+	−	−	−	−	−	−	-
SHARP	+				−								+	−	−															
STRESSED																			+	+		+				−	+	−		+

```
L
+
+
+
0
0
0
0
0
0
0
0
0


L / Y < @ %   = * # 0 ,
+ + + + + +   = * # 0 ,
+ - - - - -         -
+ - - - - -         -
+ - - - - -         -
- + + - - -         +
- - - - 2 +         -
- 2 + + + -         -
- - - - - -         -
- - - - - -         -
+ + + + + +         -
+ + + + + +         +
- - - - - -         -
                      +
  - + -   +
```

FEATU

CARD

```
 *JA *JEXAL N*A=TR*CJKE IZ=G*ORODA TIFL*ISA #
                      * J A   * J E X A L   N * A = T R * C J K E   I Z = G * O
VOCALIC               * - +   * - + - + +   - * + = - + * + - - +   + - = - * +
CONSONANTAL             - -     - - + - +   +   -   + +   - - + -   - +   +   -
ANTERIOR                - -     - - - - +   +   -   + -   - - - -   - +   -   -
CORONAL                 - -     - - - - +   +   -   + +   - - - -   - +   -   -
HIGH                    + -     + - + - -   -   -   - -   - + + -   + -   +   -
LOW                     - +     - - - + -   -   +   - -   - - - -   - -   -   -
BACK                    - +     - - + + -   -   +   - -   + - + -   - -   +   +
STRIDENT                - -     - - - - -   -   -   - -   - - - -   - +   -   -
NASAL                   - -     - - - - -   +   -   - -   - - - -   - -   -   -
CONTINUANT              - +     - + + + +   -   +   - +   + - - +   + +   -   +
VOICED                  + +     + + - + +   +   +   - +   + + - +   + +   +   +
ROUND                   - -     - - - - -   -   -   - -   + - - -   - -   -   +
SHARP                   +       +                           +
STRESSED                          +                       +     +   +         +

                      * J A   * J E X < L   N * A = T R * C J K, /   I Z = G *
VOCALIC               * - +   * - + - + +   - * + = - + * + - -   +   + - = - *
CONSONANTAL             - -     - - + - +   +   -   + +   - - +   -   - +   +
ANTERIOR                - -     - - - - +   +   -   + -   - - -   -   - +   -
CORONAL                 - -     - - - - +   +   -   + +   - - -   -   - +   -
HIGH                    + -     + - + - -   -   -   - -   - + +   +   + -   +
LOW                     - +     - - - - -   -   +   - -   - - -   -   - -   -
BACK                    - +     - - + + -   -   +   - -   + - +   2   - -   +
STRIDENT                - -     - - - - -   -   -   - -   - - -   -   - +   -
NASAL                   - -     - - - - -   +   -   - -   - - -   -   - -   -
CONTINUANT              - +     - + + + +   -   +   - +   + - -   +   + +   -
VOICED                  + +     + + - + +   +   +   - +   + + -   +   + +   +
ROUND                   - -     - - - - -   -   -   - -   + - -   -   - -   -
SHARP                   +       +                   -       + +            -
STRESSED                          +                       +      +   +
```

FEATURE

```
I Z = G * O R O D A   T I F L * I S A   #
+ - = - * + + + - +   - + - + * + - +   #
- +   +   - + - + -   + - + +   - + -
- +   -   - - - + -   + - + +   - + -
- +   -   - + - + -   + - - +   - + -
+ -   +   - - - - -   - + - -   + - -
- -   -   - - - - +   - - - -   - - +
- -   +   + - + - +   - - - -   - - +
- +   -   - - - - -   - - + -   - + -
- -   -   - - - - -   - - - -   - - -
+ +   -   + + + - +   - + + +   + + +
+ +   +   + + + + +   - + - +   + - +
- -   -   + - + - -   - - - -   - - -

+         +   +         +       +

I Z = G * O R < D <   T, / F, L, * I S <   #
+ - = - * + + + - +   -  + -  +  * + - +   #
- +   +   - + - + -   +  - +  +    - + -
- +   -   - - - + -   +  - +  +    - + -
- +   -   - + - + -   +  - -  +    - + -
+ -   +   - - - - -   -  + -  -    + - -
- -   -   - - - - -   -  - -  -    - - -
- -   +   + - + - +   -  2 -  -    - - +
- +   -   - - - - -   -  - +  -    - + -
- -   -   - - - - -   -  - -  -    - - -
+ +   -   + + + - +   -  + +  +    + + +
+ +   +   + + + + +   -  + -  +    + - +
- -   -   + - - - -   -  - -  -    - - -
      -               +    +  +
+         +   +          +         +
```

END

OF

OVERSIZE

DOCUMENT(S)

BIBLIOGRAPHY

Cherry, E. Colin, Morris Halle, and Roman Jakobson. "Toward the Logical Description of Languages in Their Phonemic Aspect," *Language*, XXIX(1953), 34-46.

Chomsky, Noam and Morris Halle. *The Sound Pattern of English*. New York, 1968.

Gross, Maurice and Andre Lentin. *Introduction to Formal Grammars*, trans. M. Salkoff. New York, 1970.

Halle, Morris. "Phonology in Generative Grammar," *Word*, XVIII (1962), 54-75.

————. *The Sound Pattern of Russian*. The Hague, 1959.

Harms, Robert T. *Introduction to Phonological Theory*. Englewood Cliffs, New Jersey, 1968.

Jakobson, Roman, Gunnar Fant, and Morris Halle. *Preliminaries to Speech Analysis*. Cambridge, Massachusetts, 1963.

Jakobson, Roman and Morris Halle. *Fundamentals of Language*. The Hague, 1956.

Wilson, Robert D. "A Criticism of Distinctive Features," *Journal of Linguistics*, II(1966), 195-206.

AN ALGORITHM FOR THE AUTOMATIC PHONETIC TRANSCRIPTION OF RUSSIAN

by

MICHAEL JOHN McCORMICK

B. A., Kansas State University, 1969

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF ARTS

Department of Modern Languages

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1971

The algorithm in this paper was written in the PL/1 computer programming language, for use on the IBM 360 computer. As such it is a working computer program. The algorithm was also written according to the transformational-generative theory of phonology, and as such it clearly reflects each stage of this transformational-generative theory as a working model.

The algorithm has as its input a distinctive feature matrix of Russian phonology and a Russian text on punched computer cards. The algorithm consists of ordered formalized rules which direct the computation as it fills the basic reference feature matrix, establishes a second matrix to accomodate the text, assigns appropriate feature specifications to the segments of the actual text, alters these specifications through the application of context-sensitive rules, and finally prints out the final phonetic transcription. This final output is a computer print-out, with phonetic symbols and complete feature specifications comprising the phonetic transcription.