BCL

REPORT NO.	7.4

TITLE:	MACHINE CODED SPEECH
AUTHOR:	John Schill
DATE:	1 December 1968
SPONSOR:	AFOSR 7-67,AF-3890 NASA NGR

BIOLOGICAL COMPUTER LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING, UNIVERSITY OF ILLINOIS, URBANA, ILLINOIS

AUTOMATIC DETERMINATION OF INVARIANCE IN MACHINE CODED SPEECH

by

John Schill
Department of Electrical Engineering
University of Illinois

BCL Report 7.4

December 1, 1968

This work was jointly sponsored by

U.S. Air Force Office of Scientific Research
AF-AFOSR 7-67

U. S. Air Force Systems Engineering Group
AF 33(615)-3890

National Aeronautics and Space Administration
Grant NGR 14-005-111

BIOLOGICAL COMPUTER LABORATORY
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITY OF ILLINOIS
URBANA, ILLINOIS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION

1.1 General Remarks

One of the major problems in speech recognition is the excessive complexity of the signal that is to be analyzed. Each syllable of a spoken word may involve several motions of the articulatory tract to generate the desired sound. Since each of these motions must produce a different sound, a speech analyzer must produce a corresponding differentiation in its output for each of these different sounds. Hence much data are produced for even a simple sentence. A method for handling these data conveniently and for presenting them efficiently was necessary. With the collection of these data made, a method of analysis to determine uniquely the invariant patterns for each spoken word had to be found. Such a method for data collection and analysis of speech will be described in the following pages.

1.2 Analysis and Synthesis: Early Developments

Speech is usually accepted as the most natural method for the transmission of information for man, and the history of "talking automata" goes very far back, indeed. "Speaking machines" developed in the eighteenth century can still be seen in museums. The most remarkable one was built by

Wolfgang von Kempelen.[5] His automaton utilized reeds to excite hand varied resonators to produce voiced sounds, while bellows provided air to the reeds. Restricted passages controlled by the operator produced the consonants. Von Kempelen's machine utilized the basic principle of modern vocoders, i.e., separating the "voicing sound generator" with little information from a "controller" with almost all the information.

The first systematic study of speech sounds, however, was done by Helmholtz in the nineteenth century.[8] He was able to synthesize vowel sounds by vibrating a number of pitchforks at selected frequencies. The first recording of the waveforms of speech sounds was made by a device called the "phonautograph." Up to the early twenties, many investigators of speech suspected that speech sounds are characterized by multiple resonances. At that time, Sir Richard Paget[13] determined that vowels and other sounds were characterized by two or three dominant resonances in the vocal tract.

1.3 Analysis and Synthesis: Recent Developments

After these promising beginnings, speech in the last few decades has been the subject of intensive research effort as a result of recent developments in the theory of

communication and artificial intelligence. The goal of this machine-oriented speech research is an effective man-machine verbal communications system. This system must convert speech into a code that permits machine recognition of speech, and ultimately must be able to synthesize speech. The first step toward this goal was Dudley's "vocoder."[5]

A vocoder is a bandwidth compression system which analyzes and later resynthesizes speech from transmitted control signals. Gazdag says, "The basic idea, from an articulatory point of view, is to transmit only signals to specify the movements of the vocal tract for the purpose of controlling an artificial vocal tract at the receiving end."[7]

The first vocoder utilized a set of bandpass filters of a bandwidth of 150 Hz. The rectified and smoothed filter outputs were modulated to synthesize the sound. A summation of the modulated control signals formed the resynthesized speech which was fairly intelligible but lacked naturalness. It was found that the transmission of the control signals required only 10 per cent of the bandwidth covered by the filters which in theory meant that only 10 per cent of the channel capacity is required to transmit the speech signals.

Since the first vocoder, there has been, despite many efforts, no major breakthrough in vocoding techniques. There have been, however, improvements in circuitry and components, but the basic principles have not appreciably changed.

1.4 Short-time Spectrum

The area of speech recognition is quite allied to that of the vocoders as the output of the filters of the vocoder do yield enough information for one to be able to understand what is being spoken into the filters. The collection of all the instantaneous filter outputs is referred to as the "short-time spectrum." Visually displayed, these spectra are clearly distinguishable and can be read with reasonable accuracy (see Figure 1 for example).

It has been noted that the "short-time spectrum" has peaks at various frequencies, known as "formant frequencies." These peaks seem to occur more or less periodically. The set of peaks at the lowest frequencies is known as the first formant; the next lowest, the second, etc. In 1952, using the first three formants, David Biddulph and Balashek[3] were able to recognize the spoken digits, "one," "two,"..., "ten," with nearly 98 per cent accuracy for a system adjusted to the speaker's voice.

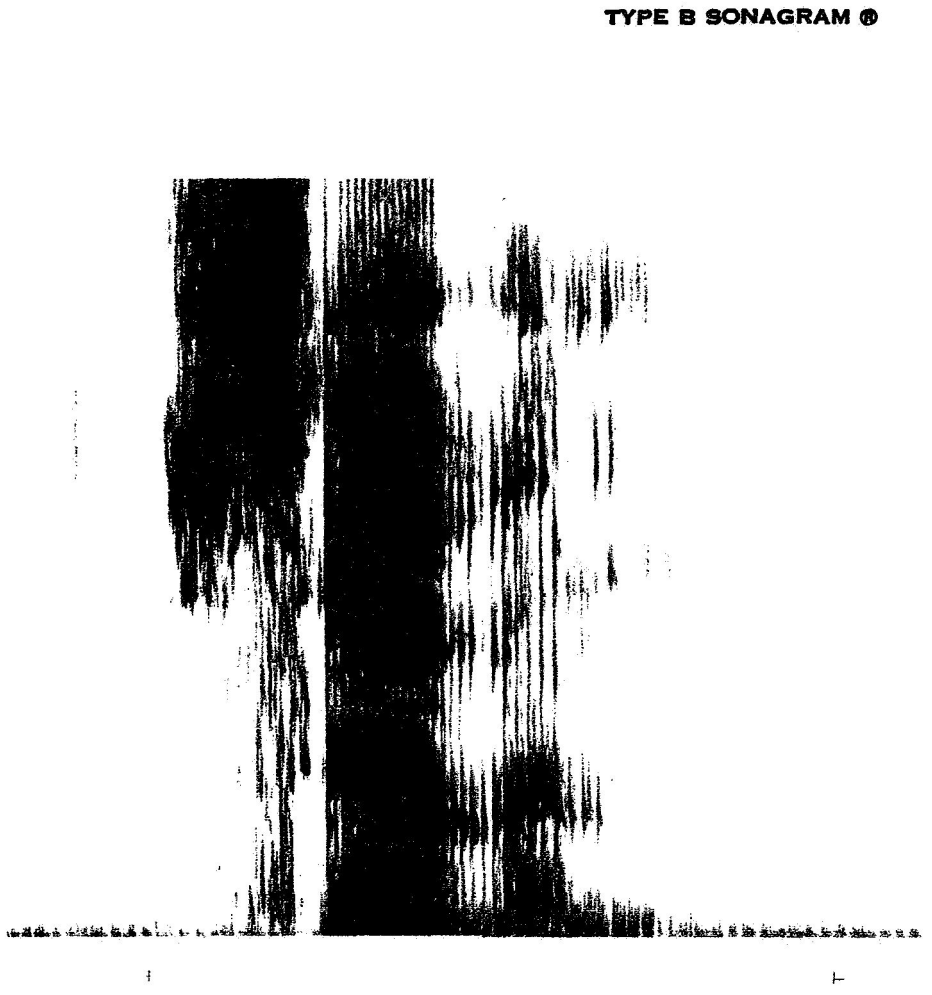


Figure 1. A Sonagram for the Word "SEVEN"

There have been many other systems since this, but all were limited to the recognition of vowels in the well defined context of the ten spoken digits pronounced under exact laboratory conditions.

The difficulties in speech recognition are quite significant. Speech by its very nature is not a "nice" continuous signal amenable to the basic tools of an engineer. Everyone speaks differently; moreover, each person's voice changes with various situations. The code of speech is not simple to break.

1.5 Gazdag's Decoder

Gazdag[7] has attacked this problem in a rather interesting manner. He challenged the usefulness of phonetic concepts as, e.g., "phonemes" in speech analysis based on mechanical processors. Instead of using phonemes as the basic speech units, he suggested letting the design of the analyzing system define these basic units which he called "machine events." Machine events are in his specific case a function of the set of output signals of the filters of a well defined system he called "Processor." Some phonemes may be machine events but others may be a series of machine events. The definition of a machine event is characterized

by the structure and operation of the machine used. Machine events are functionally defined. They yield information about the articulation of the speaker in order to differentiate conveniently his speech sounds.

Gazdag gives two requirements for such a Processor. The Processor must operate invariantly with respect to various intensities of like utterances; secondly, it must operate invariantly with respect to temporal variations of like utterances. The reasons for these requirements are fairly obvious. Intensity invariance is needed because one usually does not speak at the same volume constantly; similarly, invariance with respect to time variation is required since one does not always speak at the same rate.

Let us consider the Processor as a part of a larger system--the Decoder. When a written message is read by a speaker into the input of Decoder, a reproduction of the written message appears at the output of the Decoder. In a functional sense, the Decoder can be thought of as a system that performs some kind of a many-to-one mapping from the signal space to the message space. The system's input is a continuously varying speech signal, while its output can assume only a finite number of discrete states. The message is expressed in terms of these output states.

Two major operations are performed by the Decoder (Figure 2). The first stage is called the "Processor." It transforms the speech signal into machine events as described above. The second stage is called the "Translator." This stage operates on the output of the "Processor," translating sequences of machine events into output messages.

First, let us consider the Processor. It has only one input channel but has n binary output channels,

$$u_i; i=1,....,n$$

where the values that the channel's signal u_i may assume are

$$u_i = 0, 1$$

At any instant, the set of outputs of the Processor will be interpreted as a row vector.

$$U = (u_1, u_2,..., u_n)$$

which can assume 2^n possible states. Each of these states signifies a machine event which in turn may be thought of as the machine representation of a set of articulatory positions of a speaker's vocal tract. At this point, it is necessary to postulate that when a speaker generates a word, his articulators move simultaneously and continuously to produce a sequence of articulatory positions. Therefore, when the speaker utters a word (or part of it) at slower or faster rates, he executes approximately the same sequence

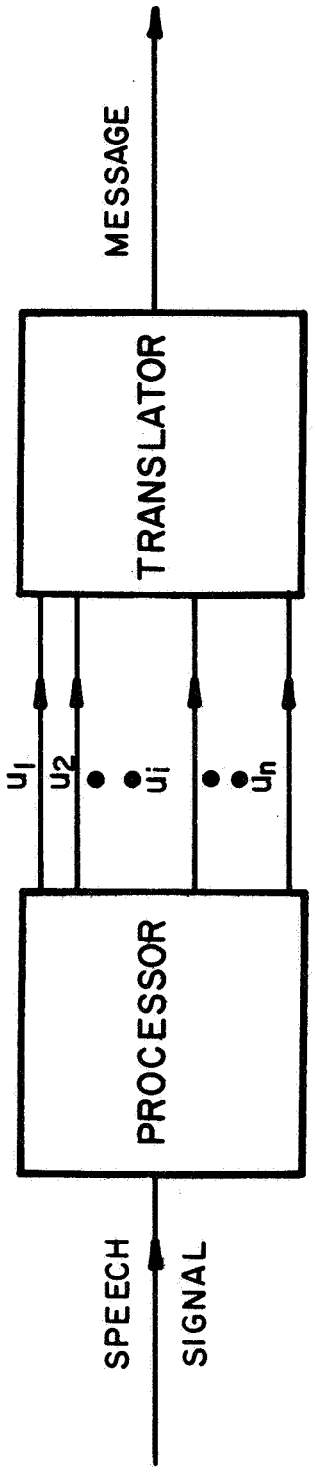


Figure 2. Gazdag's Decoder.

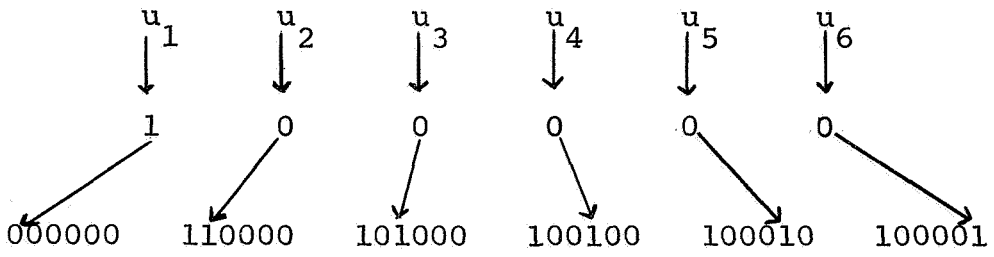
of articulatory gestures at a slower or faster rate. The necessity of this postulate is obvious from the viewpoint of parsimony as well as common sense. Even if we spoke using different articulatory positions at different rates while saying the same word, the sequence of sounds we hear must have considerable similarity; otherwise the different utterances would not sound as though they represented the same word. Hence the information about the identity of a spoken word is represented by the order of occurrence of a sequence of N machine events.

$$U_0,\ U_1,\ U_2,\ldots,\ U_N$$

The duration of each U_j provides an indication of the rate of utterance but its duration is irrelevant as printed equivalent of the utterance.

Each u_i of the row vector represents independent properties relating to the utterances; therefore if independent in value, the probability of more than one element of the row vector changing at the same instant is infinitesimally small for practical purposes. Therefore the Hamming distance between two adjacent machine events must be unity. This means that only one component, u_i, of two adjacent row vectors changes its value. For example, consider the

following row vector:



It can change only to the row vectors shown above.

Let us consider what this implies for the decoding of speech. First, consider the sequence of N machine events,

$$U_0, U_1, U_2, \ldots, U_N$$

where U_i is the n-digit binary number

$$u_{i,1}, u_{i,2}, \ldots, u_{i,n}$$

Since the possible representation of each machine event is 2^n, the number of possible representations of a spoken word of a sequence of N machine events is 2^{nN} if the constraint of a Hamming distance of one unit for adjacent machine events does not hold. For example, the spoken digits, "one," "two," "three," etc., are represented by between 8 and 15 machine events, while more polysyllabic words will, of course, require many more machine events to represent them. This means that if the machine representation of a spoken word was a sequence, say of 10 machine events (N=10),

each with 6 output channels (n=6), there could be 2^{60} or approximately 10^{18} possible representations. However, taking the constraint of a Hamming distance of one unit only between subsequent machine events into consideration, the number of possible representations decreases drastically. For the representation of 10 machine events with 6 output channels as before, there are only $1.1x10^{10}$, or approximately 2^{34} possible representations. In other words, the constraint on two subsequent machine events provides a code that reduces the enthopy of unconstrained sequences of 60 bits, as in the example above, to slightly more than 34 bits.

Because of the ambiguity of the encoding process, a set of speech events for a particular spoken word by a particular speaker may vary from trial to trial. However, due to the fact that the vocal tract goes through approximately the same process for each utterance of a word, the sequences should possess many of the same machine events. Gazdag calls the sequence of common machine events a "Significant Subsequence," (SSS). This means that in repeated utterances of the same word, W

$$S_1(W);\ U_{1,0},\ U_{1,1},\ U_{1,2},\ldots,\ U_{i,j},\ldots,\ U_{1,N_1}$$

$$S_i(W);\ U_{i,0},\ U_{i,1},\ldots,\ U_{i,j},\ldots,\ U_{i,N_i}$$

$$S_r(W);\ U_{r,0},\ U_{r,1},\ldots,\ U_{r,j},\ldots,\ U_{r,N_r}$$

(where $S_i(W)$ represents the sequence of N_i machine events of the i-th utterance of word and $W_{U_{k,l}}$ represents in the kth utterance of the same word W the lth machine event in the representation of that utterance), there exists the largest common sequence of speech events that is sufficient to signify this particular word.

Let us consider the sequence $S_i(W)$. It is an ordered set of N_i elements. The number of ordered subsets of m elements for $S_i(W)$ is

$$\mathcal{N}(N_i, m) = \binom{N_i}{m}$$

The total number of non-empty subsets is

$$\mathcal{N}(N_i) = 2^{N_i} - 1$$

For each utterance $S_i(W)$, all the subsets are established: $\mathcal{N}(N_i)$ (i.e., the set of all ordered subsets). We now form the intersection of all the $SS_{(N_i)}$

$$\bigcap_i SS_{(N_i)}$$

and select the elements with largest m of this new set. If there is one and only one, this is the SSS. If there are more, we iterate.

Let us look at some of the characteristics of these SSS's. Consider the following SSS for the word W of repeated utterances.

$$U^*_1, U^*_2, \ldots, U^*_j, \ldots, U^*_N$$

where U^*_j represents the jth machine event of the SSS for the word W. The most obvious characteristic is that each machine event, U^*_j, is included in each $S_i(W)$; otherwise it could not be a characteristic part of that word W. If this were not the case, the sound of the vowel "u" could be said to be characteristic of the sound of the word "car."

The second characteristic is the sequential nature of the subsequence itself. The particular sound represented by U^*_{i-1} must be followed by, but does not necessarily have to be, immediately adjacent to, U^*_i in a particular original sequence. For example, U^*_8 may correspond to the tenth machine event, $U_{i,10}$, in a particular sequence i, while U^*_9 may correspond to the fourteenth machine event, $U_{i,14}$. These two characteristics are the two major pillars of the hypothesis which claims that a spoken word can be identified by its SSS.

Let us now briefly consider the Processor used in this experiment, for the specific details are discussed elsewhere.[7] (See Figure 3.) In general, this Processor consists of twelve filters which cover the frequency spectrum from 150 Hz to 3000 Hz. Each covers a particular band of this spectrum. The outputs of the filters, a_i, are analog in nature, thereby yielding information not only of the existence of a component at that filter's frequency but also its magnitude.

The information content of the set of these filter outputs is redundant[15], and the first problem is to find a way to reduce this redundancy. If we consider the set of all filter outputs to be represented by a twelve dimensional space, the instantaneous state of all filters may be represented by a vector in this space. After considerable experimentation, Gazdag was able to show that the essential information about speech sounds is preserved by locating this vector in well defined regions of this space that are separated by six strategically placed hyperplanes. These hyperplanes partition the twelve dimensional space into 64 regions. The vector in this twelve dimensional space which represents the particular sound entering the filters at a given instant in time moves through these 64 regions,

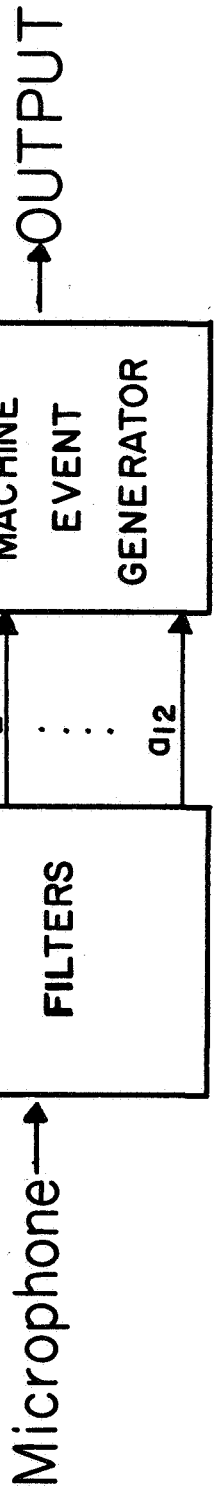


Figure 3. The Processor

causing the machine event generator (Figure 3) to have a specific output state corresponding to the region in which the vector is at that given instant. These output states are called the "machine events" which were discussed earlier. Due to the time response of the filter outputs, any machine event to be considered meaningful must persist for at least 20 msec. Using this criterion plus the one unit Hamming distance constraint, Gazdag was able to determine the SSS for a specific word, given several strip recordings from the machine event generator of repeated utterances of that word. (See Figure 4 for examples.) This procedure is tedious and time-consuming. The problem that will be dealt with in the following pages is how to define operationally this procedure of determination of these SSS's and how to assemble a system to do this determination automatically.

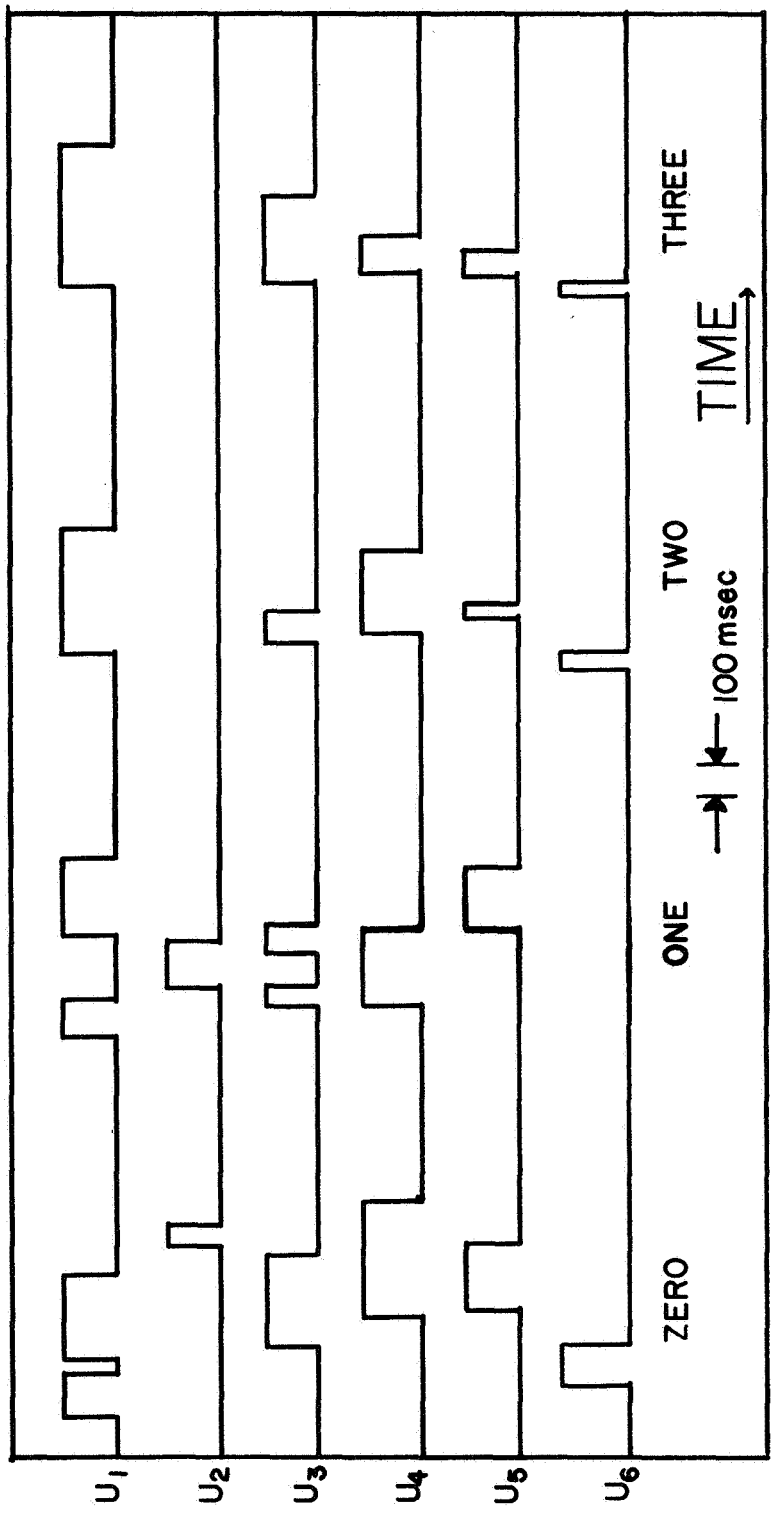


Figure 4. Examples of the Processor's Output.

2. THE ALGORITHM

Before any hardware can be made, it is necessary to find first an algorithm that computes significant subsequences, (SSS). Let us consider Table 1. Each column represents the machine encoding of a particular spoken word "W"--in this case it is the digit "ONE"--in the form of a sequence of machine events. In developing the desired algorithm, one sequence may be chosen as the standard of comparison to which the other sequences are then compared.

Let us pick the sequence S_1 as the standard. The first member of the standard, $U_{1,1}$, is compared with the members of the other sequences in the following manner. When the S_2 sequence for the comparison is used, $U_{1,1}$ is compared to $U_{2,1}$. If it is identical, the fact is noted and the next sequence, say S_3, is inspected. If $U_{1,1}$ and $U_{2,1}$ are not identical $U_{1,1}$ is compared to $U_{2,2}$. If they are identical, the fact is noted and the next sequence is inspected. If $U_{1,1}$ and $U_{2,2}$ are identical, the process is continued until a match is found or a quarter of the length of the S_2 (the LOWER SEARCH LIMIT) has been searched. If no match has been found, the fact is noted and the next sequence is searched in the same way as the S_2 sequence was

$S_1(W)$	$S_2(W)$	$S_3(W)$	$S_4(W)$	$S_5(W)$
$U_{1,1}$ = 000000	$U_{2,1}$ = 000000	$U_{3,1}$ = 000000	$U_{4,1}$ = 000000	$U_{5,1}$ = 000000
$U_{1,2}$ = 100000	100000	100000	100000	100000
101000	101000	101000	101000	101000
001000	001000	001000	001000	101100
001100	001100	000000	000000	100100
000100	000100	000100	000100	000100
010100	010100	010100	010100	010100
010110	010110	010110	010110	010110
011110	011110	011110	011110	011110
111110	111110	111110	111110	111110
101110	101110	101110	101110	101110
101010	101010	101010	101010	101010
100010	100010	100010	100010	101110
100000	100000	100000	100000	101010
$U_{1,15}$ = 000000	$U_{2,15}$ = 000000	$U_{3,15}$ = 000000	$U_{4,15}$ = 000000	100010
				100000
				$U_{5,17}$ = 000000

Table 1. Examples of the Machine Events
for Repeated Utterances of the Word "ONE"

inspected. The process continues until the sequences other than the chosen sequence have been inspected for the existence of a machine event identical to $U_{1,1}$. Once all sequences have been searched, a tally--or vote--of the number of sequences having a match for $U_{1,1}$ is taken. If all the sequences, or all but one sequence, have a match for $U_{1,1}$, then $U_{1,1}$ is considered a COMMON MACHINE EVENT. This is called the "ALL BUT ONE CONSTRAINT." In all the sequences which have a match for $U_{1,1}$ the UPPER SEARCH LIMIT is set equal to the sequential location plus one of the match $U_{1,1}$. This means that in searching for a match for $U_{1,2}$, the search will begin in a particular sequence at the upper search limit and continue until a match is found or a quarter of the length of the sequence beyond the upper search limit or the end of the sequence, whichever comes first, is inspected. In the case of a sequence without a match for $U_{1,1}$, the upper search limit is the first machine event. All common elements taken in order of discovery form the SSS.

When the member of the chosen sequence is located toward the end of the sequence, the lower search limit of the other sequence (i.e., the upper search limit plus a quarter of the sequence length) may exceed the length of that

sequence. Therefore it is necessary to redefine the lower search limit as the last member of that sequence. (See Figure 5 for the cases.)

There is also a problem if the last members of two sequences are chosen as common subevents, and there still are elements in the chosen sequence to be tested for matching. In this case, the algorithm is ended since the criterion for a common machine event cannot be satisfied.

There is one weakness with the algorithm as it now stands. Consider the following sequence of numbers representing machine events in sequences S_1, S_2,..., S_5

S_1: 1 2 3 1 4 7 6 3

S_2: 1 2 3 4 1 7 6 3

S_3: 1 2 3 5 1 4 1 7 6 3

S_4: 1 5 2 3 1 4 1 7 6 3

S_5: 1 2 3 1 2 4 1 8 7 6 3

If sequence S_1 were used as the chosen standard, the SSS is

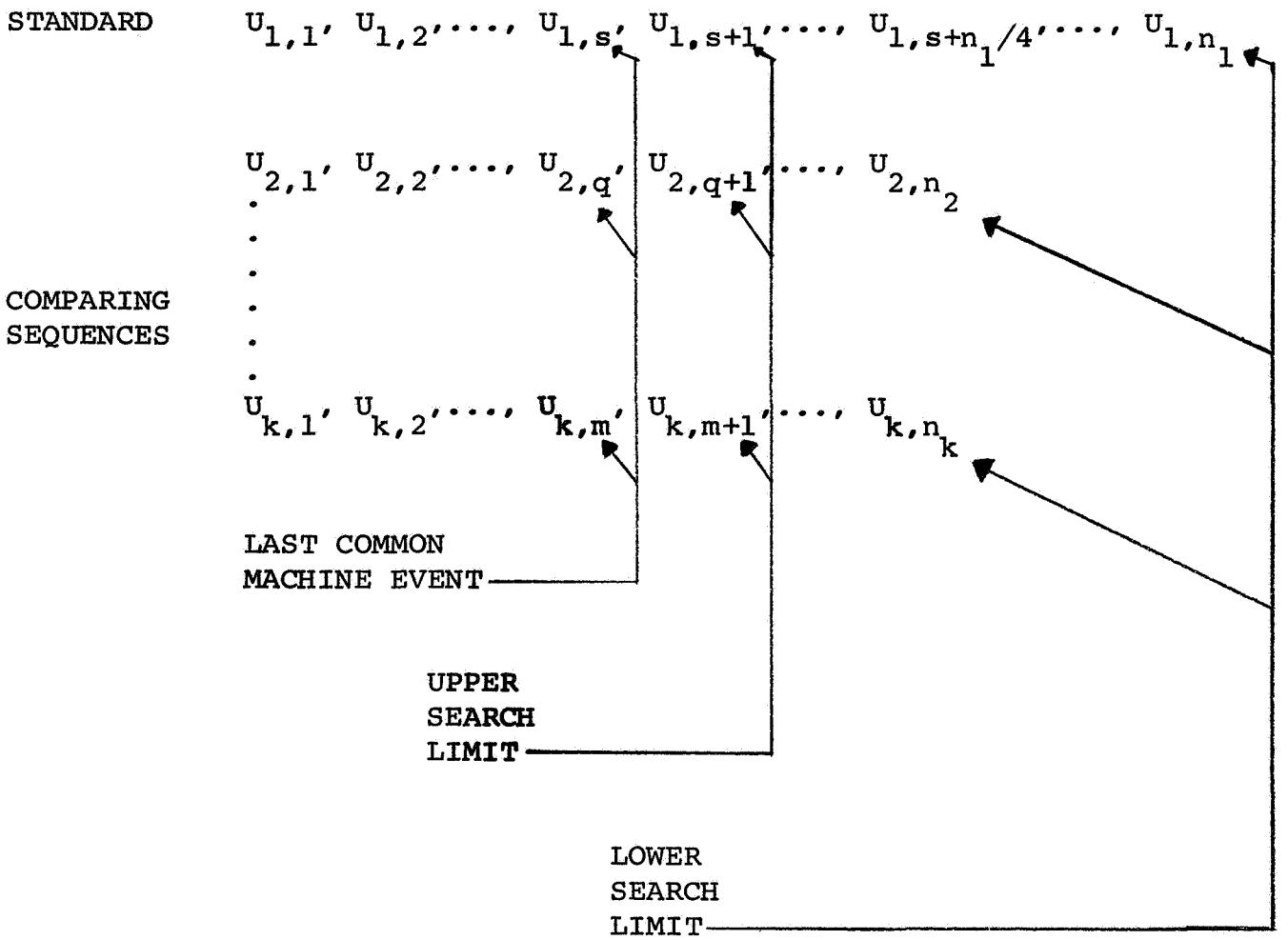
SSS_1: 1 2 3 1 4 7 6 3

If sequence S_2 were used as the chosen standard, the SSS is

SSS_2: 1 2 3 4 1 7 6 3

If any of the remaining S_i's were used as the chosen standard, the SSS is

SSS_i: 1 2 3 1 4 1 7 6 3, $i = 3, 4, 5$.



Where $U_{a,b}$ is in the a-th sequence the b-th machine event

and $s + n_1/4 < n_1$, $q + n_2/4 > n_2$, $m + n_k/4 > n_k$.

Figure 5. An Example of the Algorithm in Progress

To eliminate this ambiguity, each sequence is used as the standard, the set of SSS's generated is treated as a set of sequences and put through the algorithm again! The result of this iterated application of the algorithm produces uniquely the "True Significant Subsequence," SSS^*. For example, using the algorithm on the five sequences above, we find the following:

Using S_1 as the standard, the SSS is

1 2 3 1 4 7 6 3

Using S_2: 1 2 3 4 1 7 6 3

Using S_3: 1 2 3 1 4 1 7 6 3

Using S_4: 1 2 3 1 4 1 7 6 3

Using S_5: 1 2 3 1 4 1 7 6 3

Now when the set of SSS's (denoted by $SSS^{(1)}$) is again put through the algorithm, using the "ALL BUT ONE CONSTRAINT," the same SSS's are again obtained. In other words, $SSS^{(1)} = SSS^{(2)}$.

But since these SSS's are not identical to each other, it is necessary to tighten the "ALL BUT ONE CONSTRAINT" to a constraint for which a machine event in the SSS must occur in <u>all</u> sequences being performed upon by the algorithm. Hence in our example, using this constraint, one obtains a

unique sequence

$$SSS^*: \quad 1\ 2\ 3\ 4\ 7\ 6\ 3$$

which represents the "true" SSS, or SSS^* for this particular utterance.

For a set of sequences of short length in the example as above, it is sufficient to operate the algorithm on these sequences and also their SSS's to find the true SSS, SSS^*. For a set of more lengthy sequences, a more complex process is necessary. The original sequences, called the 0th order SSS or $SSS^{(0)}$, are operated on by the algorithm to find a set of $SSS^{(1)}$, called the first order SSS's. If all the $SSS's^{(1)}$ are identical, then the true SSS is any of these first order SSS's, i.e., $SSS^{(1)} = SSS^*$. If such is not the case, the algorithm is used on the first order SSS to get the second order SSS's, $SSS^{(2)}$. If these are not identical to each other, but identical to $SSS^{(1)}$, then the "ALL BUT ONE CONSTRAINT" is stiffened as in the example and the $SSS^{(1)}$ are again operated upon by the algorithm to find the SSS^*. If the $SSS^{(2)}$, are identical, the SSS^* is found. If neither of these cases holds, the algorithm is is used until there is found a set of identical SSS's, the nth order SSS's, which is the true SSS, i.e., SSS^*. It can be conjectured that since at each stage some of the non-

common elements are eliminated from the SSS's, the SSS* must exist. No formal proof of this conjecture will be offered here as it is beyond the scope of this paper.

To utilize this algorithm to analyze the sequences for the spoken digits, a computer program was developed, and examples of the zeroth order are given, along with the computed first and second order SSS's. They are shown in Tables 2, 3 and 4 for convenience in base 10 form by utilizing the computer program instead of in their original binary representation. For all cases tried, the spoken digits, the SSS's converged at most by the second order SSS. The data input and output and the flow chart of this program will be discussed Appendix A and B, respectively, along with a card listing.

0th Order SSS for the word "ONE"

00 32 36 44 12 04 20 28 12 44 40 42 34 32 00

00 32 36 04 20 28 12 44 40 42 34 32 00

00 32 40 44 12 04 20 28 30 14 12 44 40 32 34 32 00

00 32 36 04 12 04 20 28 12 44 40 32 34 32 00

00 32 36 44 12 04 20 28 12 44 40 42 34 32 00

1st Order SSS for the word "ONE"

00 32 36 12 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 12 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 12 04 20 28 12 44 40 34 32 00

2nd Order SSS for the word "ONE"

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

00 32 36 04 20 28 12 44 40 34 32 00

$$SSS^{*} = SSS^{(2)}$$

Table 2. The Significant Subsequences for the Word "ONE"

0th Order SSS for the word "TWO"

00 01 00 32 40 44 36 38 36 32 00

00 20 00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

00 02 00 01 00 01 00 32 40 44 36 32 00

1st Order SSS for the word "TWO"

00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

00 01 00 32 40 44 36 32 00

$$SSS^{*} = SSS^{(1)}$$

Table 3. The Significant Subsequences for the Word "TWO"

0th Order SSS for the word "THREE"

00 01 00 32 40 42 34 38 36 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 01 00 32 34 42 46 44 40 32 00

1st Order SSS for the word "THREE"

00 32 34 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

2nd Order SSS for the word "THREE"

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

00 32 34 42 46 44 40 32 00

$$SSS^{*} = SSS^{(2)}$$

Table 4. The Significant Subsequences for the Word "THREE"

3. THE SAMPLER

To connect the Processor to the computer which contains the algorithm, the author found it necessary to provide a "filter" as an interface, called the "Sampler" (second rack on the right of Gazdag's Processor; Figure 6). It eliminates most of the noise generated by the Processor as, for instance, the Processor's transient states. The sampler signals to the computer that the Processor has a valid signal. The computer functions as a storage device as well as the implementation of the algorithm.

The sampler samples the output of the Processor and transmits its states to the computer and operates in two modes: a synchronous and an asynchronous mode. In the synchronous mode the rate at which the Processor's output is sampled is controlled by an external clock pulse (CP). In the asynchronous mode the Processor is sampled continuously. In this mode, a change of the Processor's state causes a change in the sampler's output.

The input of the sampler can be of two forms: inputs compatible with integrated circuit logic levels and those which are not. For the latter, the circuit in Figure 7 converts the input level to levels necessary for operation

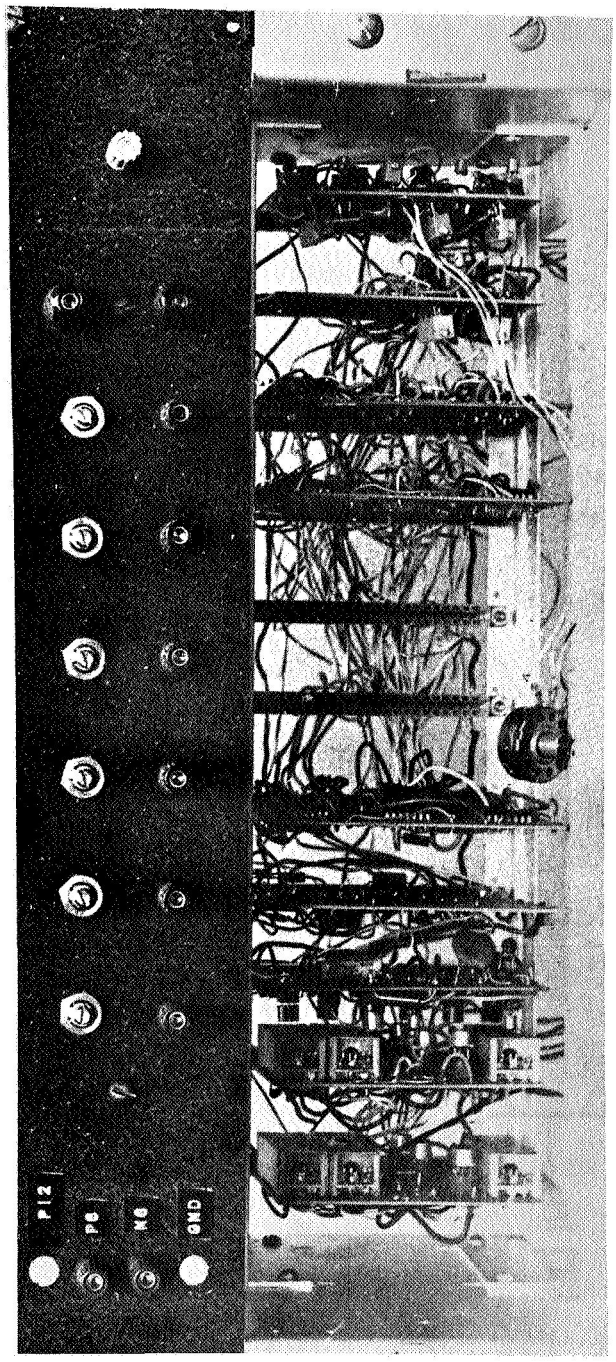


Figure 6. The Processor with the Sampler

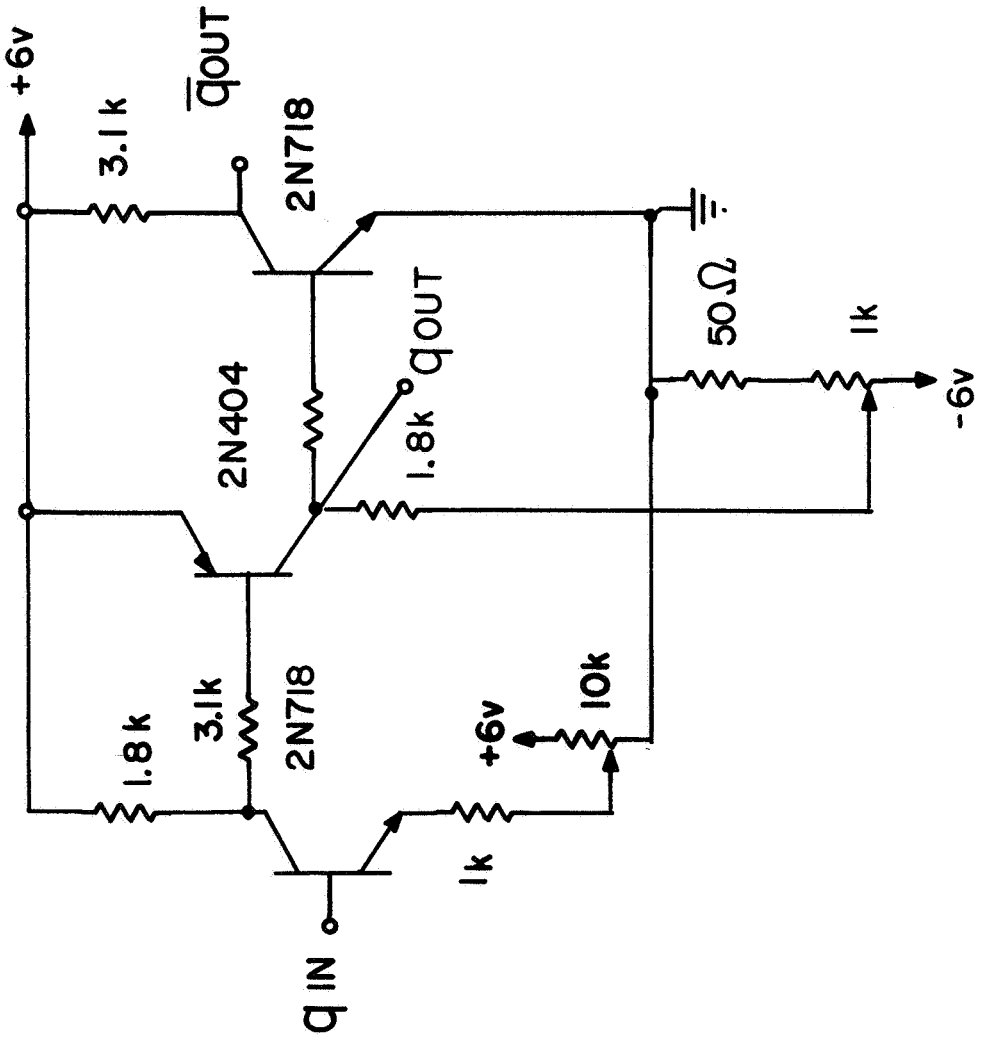


Figure 7. Incompatible Logic Level Converter

of the integrated circuits (+6 volts for high level, zero volts for low level). The potentiometer, R3, controls the input triggering level. The outputs of this circuit give both the input logic level q_i and its complement, $\bar{q}_i$. For compatible inputs the integrated circuit (IC) schematic is shown in Figure 8. The outputs of this circuit are both the input logic level and its complement.

The input logic circuit feeds into comparing circuitry (Figure 9). These inputs feed a group of JK flip-flops operating in a synchronous mode which store the PRESENT STATE, q_i, of the Processor. Another group of flip-flops in the asynchronous mode contains the LAST STATE, q_{l_i}, of the Processor. The present and last states are EXCLUSIVE or compared through a series of "NAND" gates. If the two states are different, another asynchronous flip-flop is set into the high state ($q_i = 1$), causing a one millisecond oneshot to activate. The oneshot's output is called the PASS GATE (PG). The PG allows the LAST STATE (q_{l_i}) and output flip-flops to change to the PRESENT STATE. (Not shown in Figure 8.) The output flip-flops drive the output circuitry.

The output circuitry (Figure 10) consists of a series of relay drivers with their relays. The drivers enable the

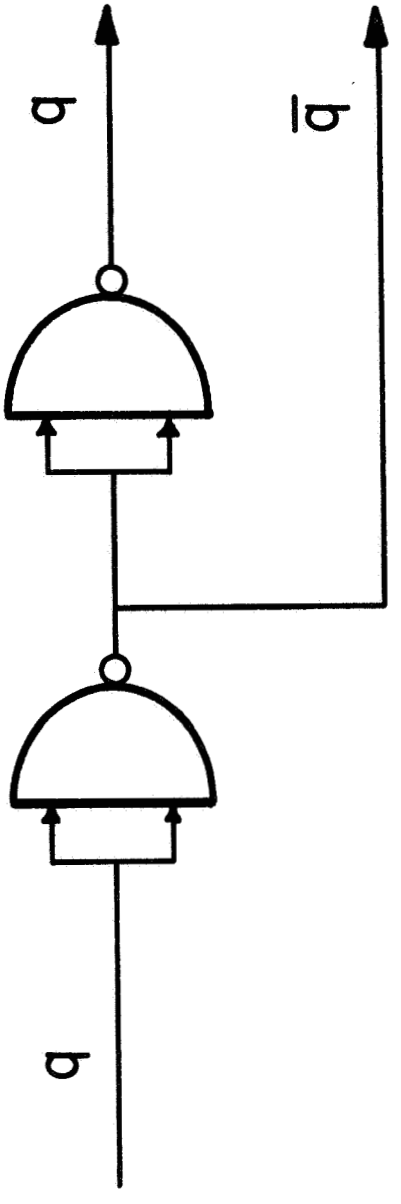


Figure 8. Compatible Logic Level Converter.

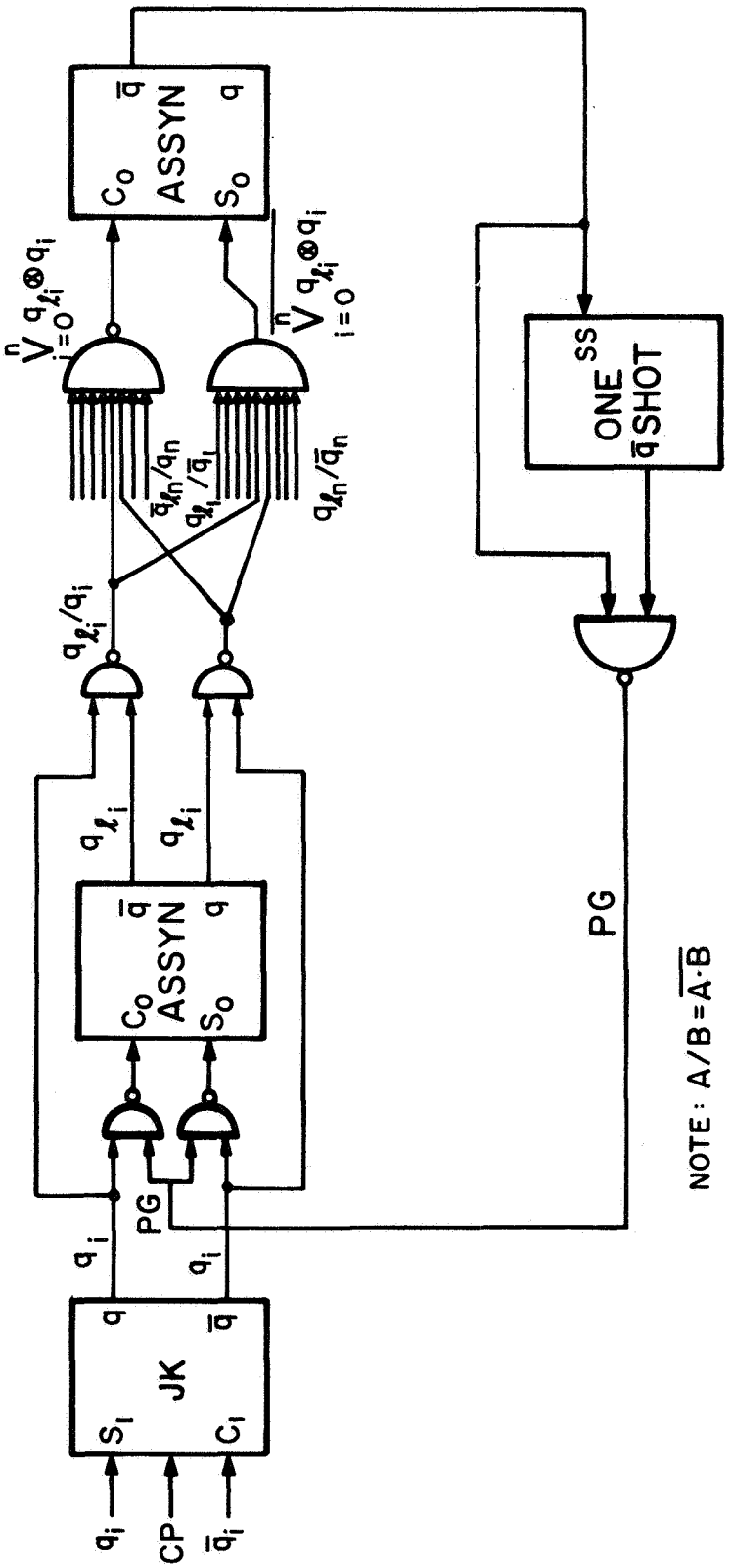


Figure 9. The Comparing Circuitry

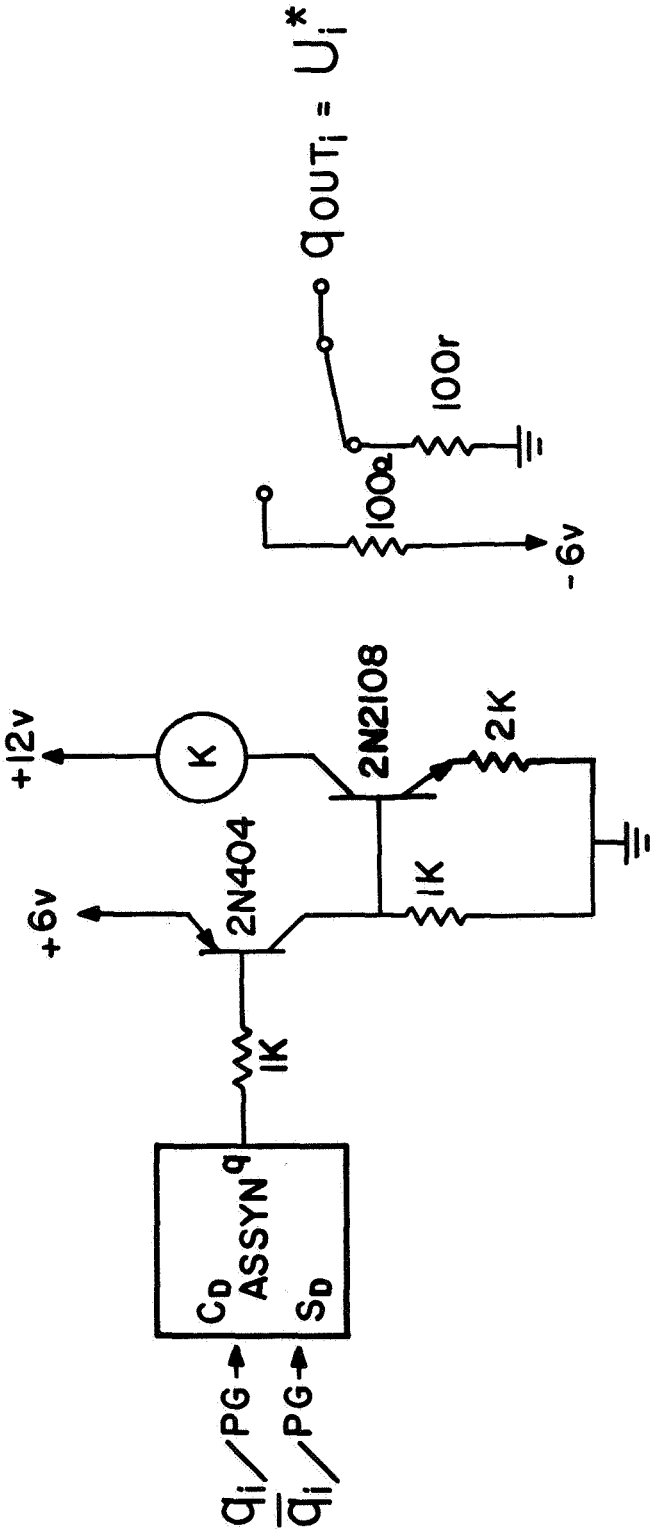


Figure 10. The Output Circuitry

IC logic to drive the relays. From the relay contacts, almost any logic level voltages can be accommodated. For the sampler to be compatible with the Illiac II, it is necessary to have logic levels of 0 (as high value) and -5V (as the low value) with 100 ohms minimum resistance. A 500 nsec oneshot is also included to pulse the Illiac II every time there is a new state (indicated by + = 1 signal). It should be noted that aside from the level compatibility, the relay response time (about 20 msec) enables the sampler to operate in an asynchronous mode without trouble from transient states.

Figure 11 shows the entire information flow through the speech recognition system. The speech signal enters the microphone of Gazdag's Processor which translates utterances into sequences of machine events. These enter the Sampler at the LOGIC LEVEL CONVERTERS and are stored (sequentially) in the PRESENT STATE flip-flops. By means of the logic circuitry of the Sampler, the valid machine events are passed sequentially to the output relays. From the contacts of these relays, the computer takes the machine events and stores them as the zeroth order SSS to be operated upon by the algorithm program. Finally, the output of the algorithm is in the form of a computer printout. The Sampler and

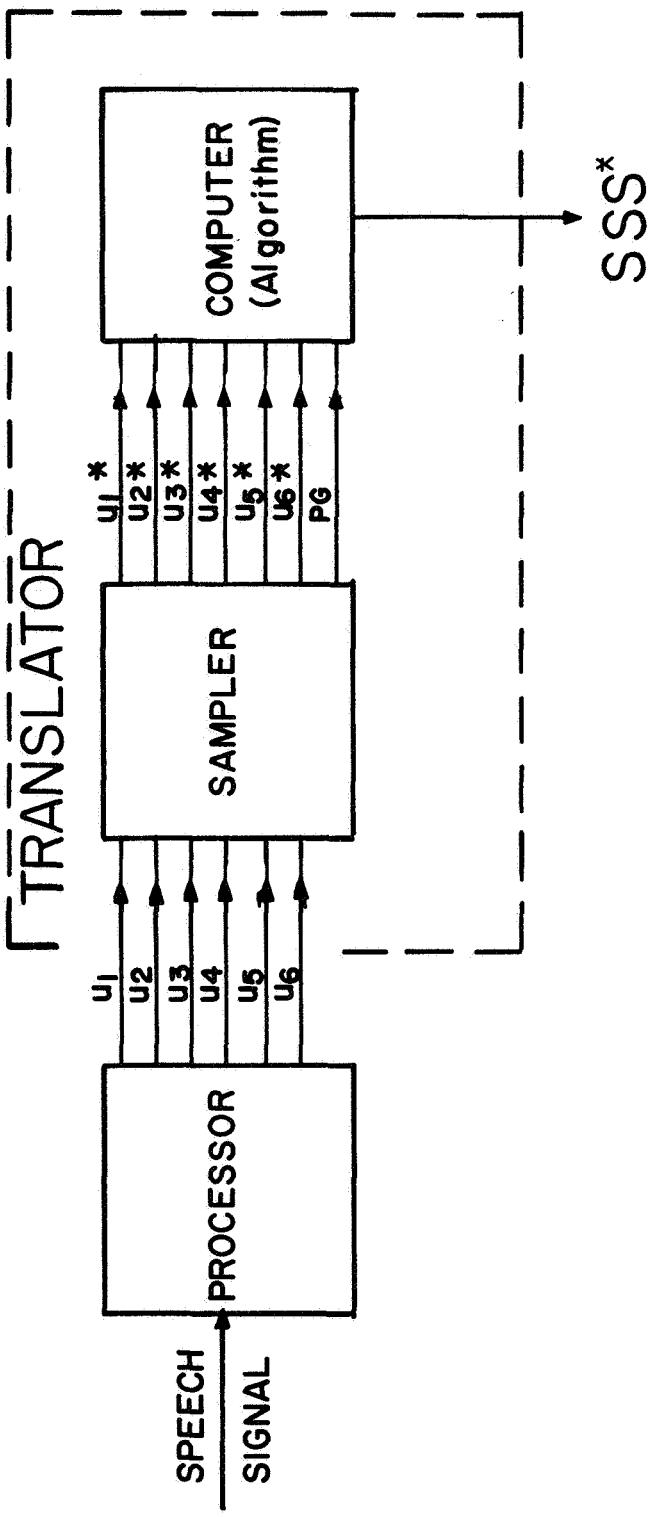


Figure 11. The Speech Recognition System

computer with the algorithm form the Translator of the Decoder. The Translator portion of the Decoder is the subject matter of this paper.

4. CONCLUSION

In this paper, an automatic method for finding invariants (SSS's) in sequences of machine events was presented.

Based on the work of Gazdag, a Sampler for his Processor was built by the author. It passes valid signals, the machine events, from the Processor to a computer. The validity of these machine events is determined by their persistence for more than 20 msec and by a Hamming distance of one unit between the adjacent events. Utilizing this Sampler, a speaker speaking into the Processor is directly connected to the computer which performs the computation of the true significant subsequences, SSS^*. This eliminates the need of manual manipulation of the recorded output of the Processor.

An algorithm was developed to find Gazdag's SSS^*. This algorithm has been implemented on the IBM 7094 and the Illiac II. Several examples of the spoken digits were tested, and indeed a true SSS was found in each case. (See Tables 2, 3 and 4 for some examples.)

With the algorithm and the Sampler it is possible to find the true SSS for words other than the spoken digits. A library of such representations of spoken words can now

be built, making it possible to achieve real-time recognition of human speech.

Although the system described here is still in a relatively simple stage, it may represent a step forward--however small--in the direction of real-time speech recognition and man-machine communication.

REFERENCES

1. Babcock, M. L. et al., A Dynamic Signal Analyzer, TR 3-1 (AF 33(616)-6428), Electrical Engineering Research Laboratory, Engineering Experimental Station, University of Illinois, Urbana (1962).

2. Crandall, L.R., On Human Communication, Science Editions, John Wiley & Sons, Inc., New York (1961).

3. Davis, K.H. et al., "Automatic Recognition of Spoken Digits," J. Acoust. Soc. Am., 24, (6), 637-642 (1962).

4. Dudley, H., "Remaking Speech," J. Acoust. Soc. Am., 2, 165 (1939).

5. Dudley, H., and T. Tarnocyz, "The Speaking Machine of Wolfgang von Kempelen," J. Acoust. Soc. Am., 22, 1951 (1950).

6. Flanagan, J.L., Speech Analysis, Synthesis and Perception, Springer-Verlag, New York (1965).

7. Gazdag, J., A Method of Decoding Speech, TR 9 (AF 7-66) Electrical Engineering Research Laboratory, Engineering Experimental Station, University of Illinois, Urbana (1966).

8. Helmholtz, H.L.F., On the Sensational Tone, Longmons, Bacon & Co., London and New York (1875).

9. Hsia, K.Y., Encoding Speech Events for Computer Recognition, M.S. Thesis, Department of Electrical Engineering, University of Illinois, Urbana (1966).

10. Judson, L.S.V., and A.T. Weaver, Voice Science, Appleton-Century-Crofts, New York (1965).

11. Millman, J., and H. Taub, Pulse, Digital and Switching Waveforms, McGraw-Hill, Englewood Cliffs, New Jersey (1965).

12. Motorola Engineering Staff, Switching Transistor Manual, Motorola, Phoenix, Arizona (1963).

13. Paget, Sir Richard, Human Speech, P. Kegan (ed.), French Trubner & Co., Ltd., London, p. 132 (1930).

14. Potter, R.K., G.A. Kopp, and H.C. Green, Visible Speech, D. van Nostrand Co., Inc., New York (1947).

15. Thomas, I., The Significance of the Second Formant, Ph.D. Thesis, Department of Electrical Engineering, University of Illinois, Urbana (1966).

APPENDIX A
Input/Output for the Algorithm Program

The input to the program can be of two forms. The first is a series of utterances of a specific word with the start of each utterance specifically identified. The second is a series of utterances with the start of each utterance unspecified. In such a case, the program looks first for events which are represented by a row vector of all zeros. These represent pauses between utterances. As soon as the next non-zero machine event is found, it is considered the start of a new utterance.

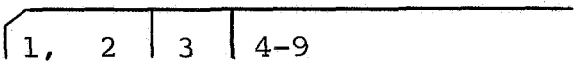
The output of the program is of printed form although is can also be in the form of punched cards. The input data are printed out as soon as they are read. Each time the program goes through the algorithm the sequences are printed out along with the significant subsequence found. There is also a diagnostic output. There are two types of diagnostic output. The first lists every comparison made with all the relevant variables. The second lists everything the first lists plus, as indicated, the passage through critical parts of the algorithm along with the associated variables.

The specific input cards are of the following form:

Card 1

I2 — The number of separate words (not utterances) to be analyzed.

Card 2 (Unseparated Utterances)

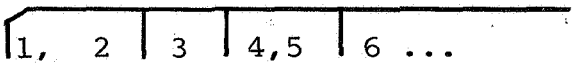


I2 — The total number of machine events in this series of utterances.

I1 — Diagnostic printout (same as on card 2 of the separated utterances).

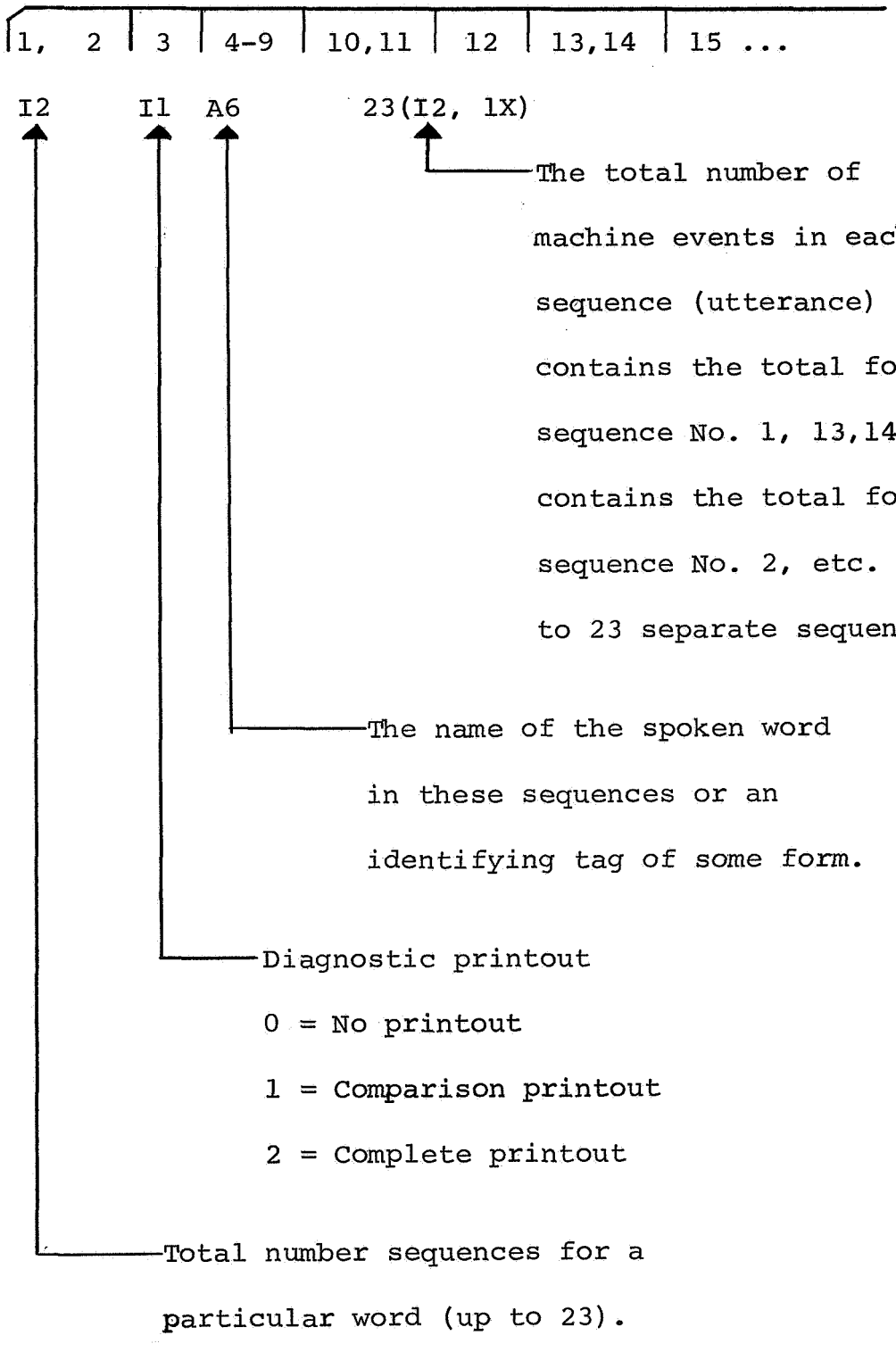
A6 — The name of the spoken word or some form of an identifying tag.

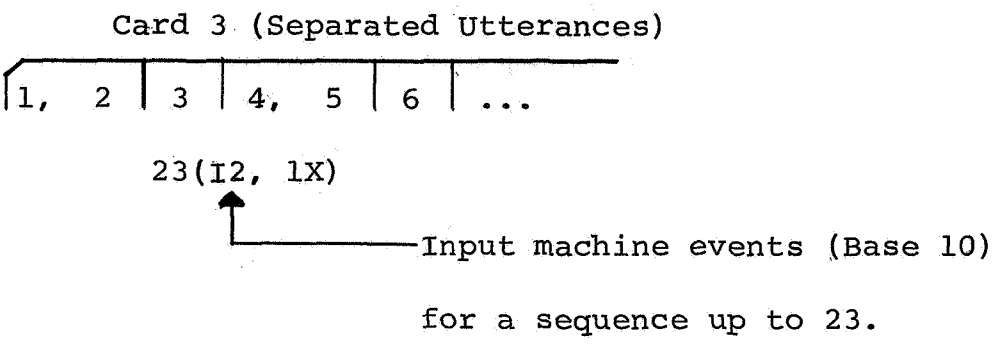
Card 3 (Unseparated Utterances)



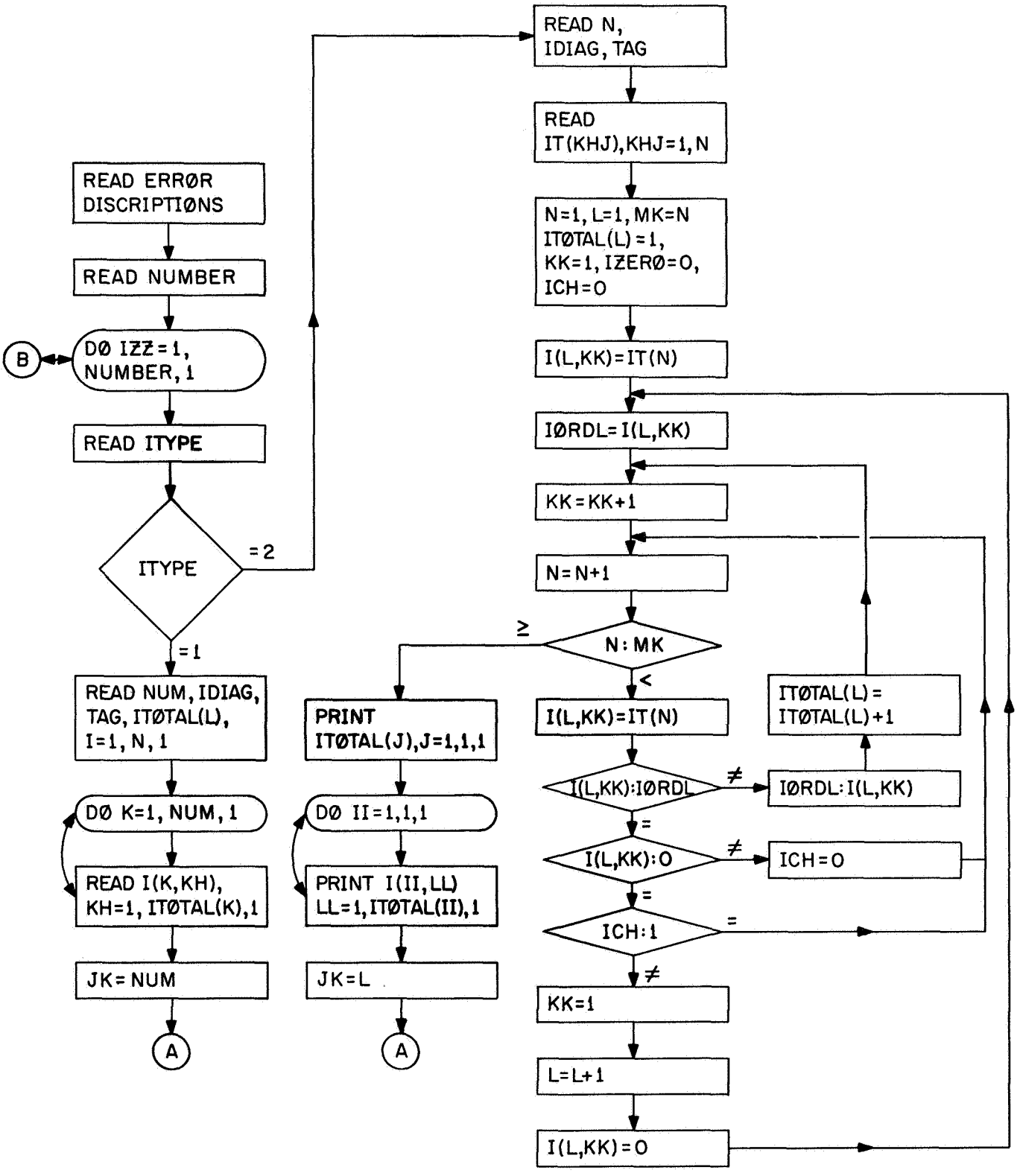
23(I2, 1X) — Input machine events (base 10) for the series of Utterances (23 per card).

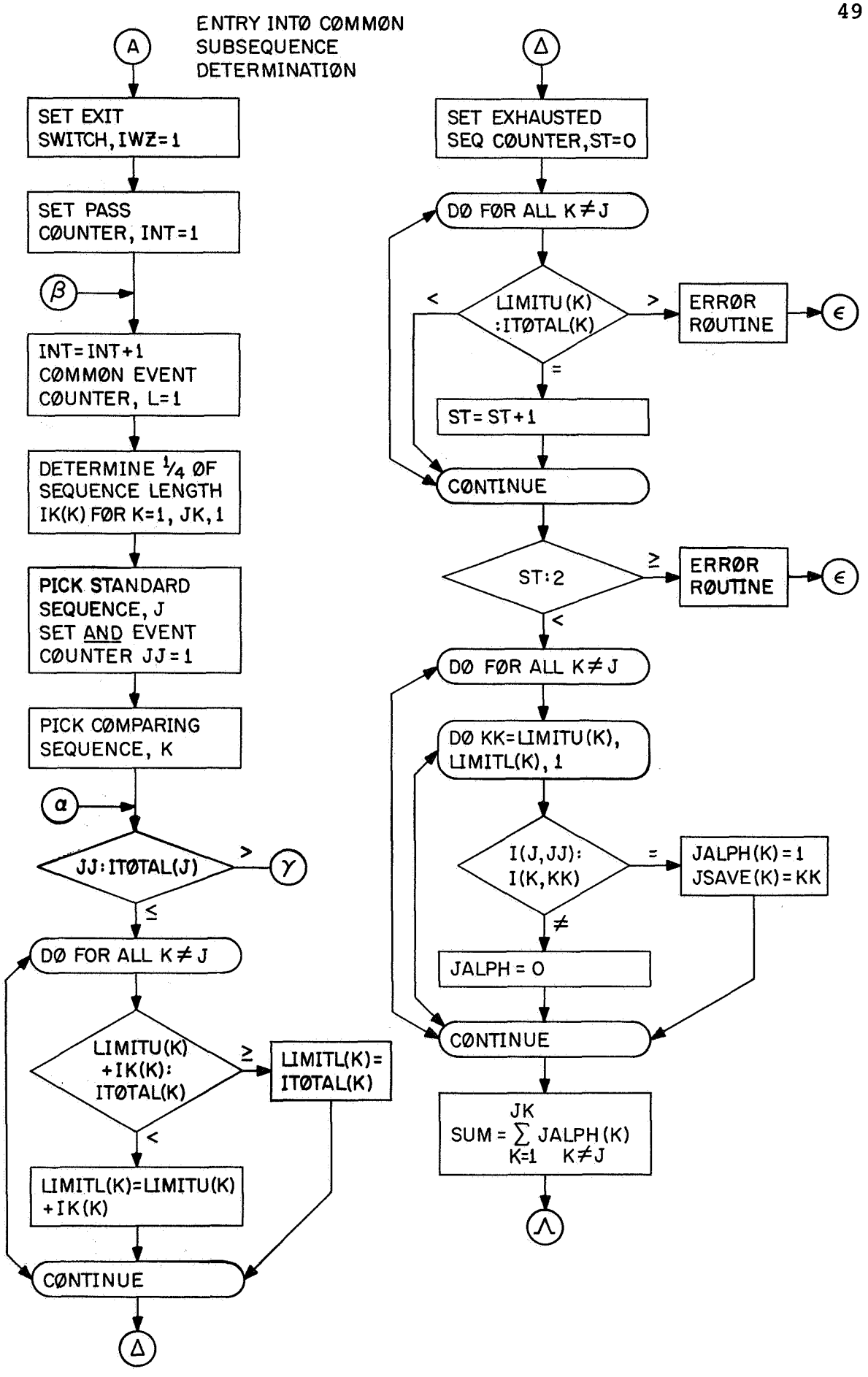
Card 2 (Separated Utterances)

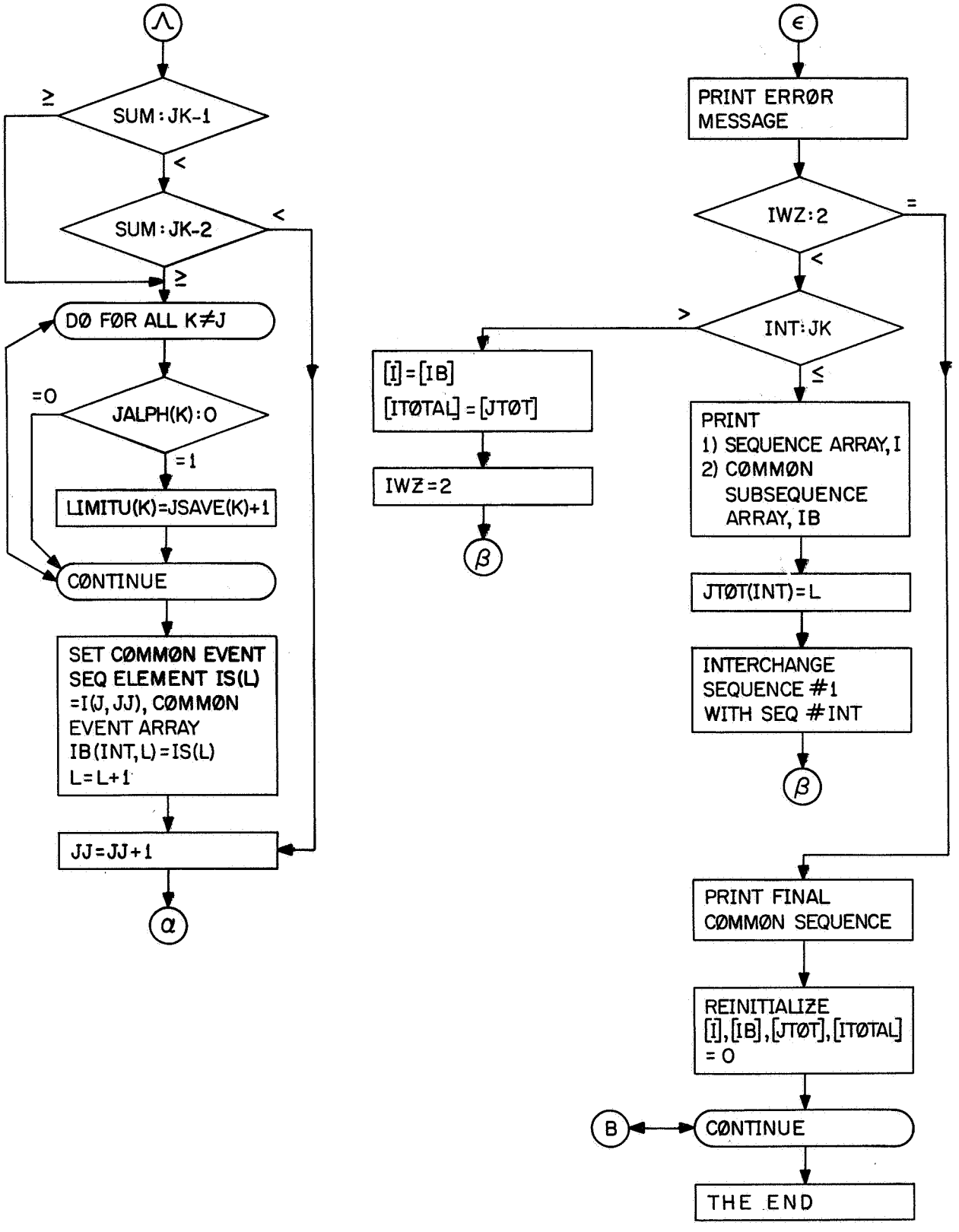



Card 3 (Separated Utterances)
1, 2 3 4, 5 6 ...
23(I2, 1X)
Input machine events (Base 10)
for a sequence up to 23.

The Program Flow Chart




ENTRY INTØ CØMMØN SUBSEQUENCE DETERMINATIØN
A
SET EXIT SWITCH, IWZ=1
SET PASS CØUNTER, INT=1
β
INT=INT+1 CØMMØN EVENT CØUNTER, L=1
DETERMINE 1/4 ØF SEQUENCE LENGTH IK(K) FØR K=1, JK, 1
PICK STANDARD SEQUENCE, J SET AND EVENT CØUNTER JJ=1
PICK CØMPARING SEQUENCE, K
α
JJ:ITØTAL(J)
>
γ
≤
DØ FOR ALL K≠J
LIMITU(K) +IK(K): ITØTAL(K)
≥
LIMITL(K)= ITØTAL(K)
<
LIMITL(K)=LIMITU(K) +IK(K)
CØNTINUE
Δ
Δ
SET EXHAUSTED SEQ CØUNTER, ST=0
DØ FØR ALL K≠J
<
LIMITU(K) :ITØTAL(K)
>
ERRØR RØUTINE
ϵ
=
ST=ST+1
CØNTINUE
ST:2
≥
ERRØR RØUTINE
ϵ
<
DØ FØR ALL K≠J
DØ KK=LIMITU(K), LIMITL(K), 1
I(J,JJ): I(K,KK)
=
JALPH(K)=1 JSAVE(K)=KK
≠
JALPH=0
CØNTINUE
SUM = JK Σ K=1 JALPH(K) K≠J
Λ


Λ
≥
SUM : JK-1
<
SUM : JK-2
<
≥
DØ FØR ALL K≠J
=0
JALPH(K) : 0
=1
LIMITU(K)=JSAVE(K)+1
CØNTINUE
SET CØMMØN EVENT SEQ ELEMENT IS(L) =I(J, JJ), CØMMØN EVENT ARRAY IB(INT, L) = IS(L)
L=L+1
JJ=JJ+1
α
[I]=[IB]
[ITØTAL]=[JTØT]
IWZ=2
β
ϵ
PRINT ERRØR MESSAGE
IWZ : 2
=
<
>
INT : JK
≤
PRINT
1) SEQUENCE ARRAY, I
2) CØMMØN SUBSEQUENCE ARRAY, IB
JTØT(INT)=L
INTERCHANGE SEQUENCE #1 WITH SEQ #INT
β
PRINT FINAL CØMMØN SEQUENCE
REINITIALIZE [I], [IB], [JTØT], [ITØTAL] = 0
B
CØNTINUE
THE END

APPENDIX C

```
$       FORTRAN
$       GO
        CALL SPR1(1)
C ITOTAL(L)     TOTAL NUMBER OF WORDS IN STRING L                          A0000101
C I(L,KK)       WORD NUMBER KK OF STRING L                                 A0000102
C L             STRING COUNTER                                             A0000103
C KK            STRING WORD COUNTER                                        A0000104
C IORDL         LAST WORD PROCESSED                                        A0000105
C IT(N)         INPUT STRING WORD N                                        A0000106
C N             INPUT STRING COUNTER                                       A0000107
C XCLOR(A,B,C) EXCLUSIVE OR OF A AND B,C=1 IF TRUE,C=-1 IF FALSE           A0000108
C ITEMP         TEMPORARY STORAGE                                          A0000109
      N4=25                                                                A0000115
   11 FORMAT(23(I2,1X),11X)                                                A0000120
   12 FORMAT(1H ,I4,40(1X,I2))
   13 FORMAT(1H1)                                                          A0000122
   14 FORMAT(1H ,14H STRING NUMBER,1X,I2,35(I2,X1))
   69 FORMAT(1H1)                                                          A0000130
   99 FORMAT(19H EXCLUSIVE OR ERROR)                                       A0000135
  804 FORMAT(1H ,15H INPUT COMPLETE)                                       A0000150
      DIMENSION ERROR(10,8)                                                A0000170
      DO 569 IJK=1,7,1                                                     A0000175
      IJL=IJK
  569 RIT 7,568,IJL,(ERROR(IJL,III),III=1,8,1)                             A0000180
  568 FORMAT(I2,8A4)                                                       A0000185
      DIMENSION JALPH(25),JSAVE(25),LIMITU(25),LIMITL(25)                  A0000190
      DIMENSION IK(25),IB(25,25),JTOT(25)                                  A0000191
      DIMENSION I(25,25),ITOTAL(25),M(25),IT(100),IS(40)                   A0000200
      COMMON I
      COMMON ITOTAL
      DIMENSION LOSER(25),TAG(2)
      DIMENSION IZJ(25,6)
C INPUT ROUTINE---(201-229)                                                A0000201
      PRINT   69                                                           A2000202
      READ   3010 ,NUMBER                                                  A2000204
      PRINT   3010,NUMBER                                                  A2000202
      PRINT   3333,NUMBER                                                   2
 3333 FORMAT(1H ,20(I5,1X))
 3010 FORMAT(26(I2,1X))                                                    A0000205
      DO 3000 IZZ=1,NUMBER,1                                               A0000206
      READ   3010,ITYPE                                                    A2000207
      IF(ITYPE-1) 3001,3001,3002                                           A0000208
C SEQUENCES ALREADY SEPARATED
 3001 READ   3011,NUM,IDIAG,TAG(1),TAG(2),(ITOTAL(LJK),LJK=1,NUM,1)        A2000209
      PRINT  3011,NUM,IDIAG,TAG(1),TAG(2),(ITOTAL(LJK),LJK=1,NUM,1)        A2000210
 3011 FORMAT(I2,I1,2A3,23(I2,1X))
      DO 3003 IIK=1,NUM,1                                                  A0000212
      KKK=ITOTAL(IIK)                                                      A0000213
      READ   11,(I(IIK,KHJ),KHJ=1,KKK,1)                                   A2000214
 3003 PRINT   11,(I(IIK,KHJ),KHJ=1,KKK,1)                                  A2000215
      JK=NUM                                                               A0000216
      GO TO 3004                                                           A0000217
C SEQUENCES NEED TO BE SEPARATED
 3002 READ   3011,N,IDIAG,TAG(1),TAG(2)                                    A2000218
      MK=N                                                                 A0000219
      READ   11,(IT(KHJ),KHJ=1,N,1)                                        A2000220
      PRINT   11,(IT(KHJ),KHJ=1,N,1)                                       A2000221
```

```
      PRINT   804                                                          A2000229
C SEPARATING INPUT STRINGS---(230-350)                                     A0000230
      ISWIT=1                                                              A2000230
      N=1                                                                  A0000231
      L=1                                                                  A0000232
      ITOTAL(L)=1                                                          A0000233
      KK=1                                                                 A0000234
      IZERO=0                                                              A0000235
      I(L,KK)=IT(N)                                                        A0000240
  100 IORDL=I(L,KK)                                                        A0000245
  101 KK=KK+1                                                              A0000250
  102 N=N+1                                                                A0000255
      IF(N-MK) 107,107,90                                                  A0000260
  107 I(L,KK)=IT(N)                                                        A0000265
      CALL XCLOR(I(L,KK),IORDL,IALPH)                                      A0000270
 1410 IF(IDIAG) 440,440,441                                                A0000277
  441 IKL=1410                                                             A0000278
      PRINT 12,IKL,N,L,KK,I(L,KK),IORDL,IALPH,ITOTAL(L),ISWIT
  440 IF(IALPH) 105,104,103                                                A0000280
  105 IORDL=I(L,KK)                                                        A0000285
      ITOTAL(L)=ITOTAL(L)+1                                                A0000290
      GO TO (101,155),ISWIT
  155 ISWIT=1
      GO TO 101                                                            A0000295
  103 CALL XCLOR(I(L,KK),IZERO,IALPH)                                      A0000300
      IKL=105
      PRINT 12,IKL,N,L,KK,I(L,KK),IORDL,IALPH,ITOTAL(L),ISWIT
      IF(IALPH) 122,104,106                                                A0000305
  106 ITEMP=I(L,KK)                                                        A0000310
      GO TO (120,102),ISWIT                                                A0000314
  120 I(L,KK)=0                                                            A0000315
      ISWIT =2                                                             A0000319
      KK=1                                                                 A0000320
      L=L+1                                                                A0000325
      ITOTAL(L)=1                                                          A0000330
      I(L,KK)=ITEMP                                                        A0000335
      GO TO 100                                                            A0000340
  122 ISWIT=1                                                              A0000341
      GO TO 105
  104 WOT 6,99                                                             A0000345
      GO TO 90                                                             A0000350
C SORTED STRINGS OUTPUT (400-                                              A0000400
   90 CONTINUE                                                             A0000405
      PRINT   12,L                                                         A2000410
      PRINT   12,(ITOTAL(II),II=1,L,1)                                     A2000415
      DO 91 K=1,L,1                                                        A0000420
      LLL=ITOTAL(K)                                                        A0000425
   91 PRINT   12,K,(I(K,J),J=1,LLL,1)                                      A2000430
      PRINT   11,(IT(KHJ),KHJ=1,N,1)                                       A2000433
C LIMITU       THE UPPER LIMIT OF COMPARISION RANGE(THE LOWEST WORD NO.)A0001000
C LIMITL       THE LOWER LIMIT OF COMPARISION RANGE(THE HIGHEST WD.NO.) A0001001
C J            SELECTED STRING NUMBER                                    A0001002
C JJ           SELECTED STRING WORD COUNTER                              A0001003
C K            COMPARING STRING COUNTER                                  A0001004
C KK           COMPARING STRING WORD COUNTER                             A0001005
C  IWZ        SWITCH TO END AFTER ALL THE REARRANGEMENTS=U,CONT,=2,EXIT
C M(L)          THE COMMON WORD ARRAY                                    A0001006
```

```
C L             COMMON WORD COUNTER                                                A0001007
C JK            THE TOTAL NUMBER OF STRINGS                                        A0001008
C INTERNAL COMPARE AND VOTE                                                        A0010200
C JALPH(K)      VOTE OF STRING K                                                   A0010201
C JSAVE(K)      SAVED POSITION LOCATION OF SELECTED WORD IN STRING K               A0010202
C JS(L)         SELECTED WORD L                                                    A0010203
C II            VOTE TOTAL                                                         A0010204
C INT             PASS COUNTER
      JK=L                                                                         A0010499
 3004 CONTINUE                                                                     A0010500
      IWZ=1                                                                        A0010501
      PRINT 69
      INT=0
 3300 CONTINUE
      INT=INT+1                                                                    A0010498
      L=1                                                                          A0010501
      DO 900 K=1,JK,1                                                              A0010520
  900 LIMITU(K)=1                                                                  A0010530
C SET SEARCH LIMITS                                                                A0010531
      DO 800 K=1,JK,1                                                              A0010532
      ICHECK=8                                                                     A0010533
      IK(K)=1.                                                                     A0010534
  803 IF(ITOTAL(K)-ICHECK) 801,801,802                                             A0010535
  802 IK(K)=IK(K)+1                                                                A0010536
      ICHECK=ICHECK+8                                                              A0010537
      GO TO 803                                                                    A0010538
  801 IK(K)=IK(K)+1                                                                A0010539
  800 CONTINUE                                                                     A0010540
      KK=1                                                                         A0010545
      K=2                                                                          A0010550
      J=1                                                                          A0010560
      JJ=1                                                                         A0010570
  990 IF(JJ-ITOTAL(J)) 999,999,2999                                                A0010580
  999 DO 901 K=2,JK,1                                                              A0010590
  902 IF(JJ+IK(K)-ITOTAL(K)) 904,904,903                                           A0010600
  904 LIMITL(K)=JJ+IK(K)                                                           A0010610
      GO TO 901                                                                    A0010620
  903 LIMITL(K)=ITOTAL(K)                                                          A0010630
  901 CONTINUE                                                                     A0010640
      ST=0.                                                                        A0010650
      DO 996 K=2,JK,1                                                              A0010660
      IF(LIMITU(K)-ITOTAL(K)) 907,908,906                                          A0010670
  906 IJK=5                                                                        A0010680
      GO TO 998                                                                    A0010690
  907 IF(IDIAG    ) 400,400,401                                                    A0010697
  401 IKL=907                                                                      A0010698
      PRINT 12, IKL,J,JJ,I(J,JJ),K,KK,I(K,KK),JK,LIMITU(K),LIMITL(K)               A2010699
  400 IF(JJ-IK(K)) 909,912,912                                                     A0010700
      GO TO 102                                                                    A0000342
  909 IF(LIMITU(K)) 910,912,912                                                    A0010710
  910 IJK=6                                                                        A0010720
      GO TO 996                                                                    A0010730
  908 ST=ST+1.                                                                     A0010740
  912 IF(ST-2.) 916,915,915                                                        A0010750
  996 CONTINUE                                                                     A0010770
  905 K=JK                                                                         A0010760
      IF(ST)  916,915,915                                                          A0010780
```

```
  915 IJK=7                                                              A0010790
      GO TO 998                                                          A0010800
  916 K=2                                                                A0010810
C THIS DETERMINES IF LIMITU(K) IS TOO LOW AND INITIALIZES LIMITU(2)       A0010999
      GO TO 1089                                                         A0011000
 1000 CALL XCLOR(I(J,JJ),I(K,KK),IALPH)                                  A0011010
 1400 IF(IDIAG) 430,430,431                                              A0011017
  431 IKL=1000                                                           A0011018
      PRINT 12,IKL,J,JJ,I(J,JJ),K,KK,I(K,KK),LIMITU(K),LIMITL(K),JK      A2011019
  430 IF(IALPH) 1002,1011,1001                                           A0011020
 1011 IJK=3                                                              A0011030
      GO TO 998                                                          A0011040
 1001 JALPH(K)=1                                                         A0011050
      JSAVE(K)=KK                                                        A0011060
      GO TO 1005                                                         A0011070
 1002 KK=KK+1                                                            A0011080
 1003 IF(  IDIAG      ) 410,410,411                                      A0011087
  411 IKL=1003                                                           A0011088
      PRINT 12, IKL,J,JJ,I(J,JJ),K,KK,I(K,KK),JK,LIMITU(K),LIMITL(K)    A2011089
  410 IF(KK-LIMITL(K)) 1000,1000,1004                                    A0011090
 1004 JALPH(K)=0                                                         A0011100
 1005 IF(K-JK) 1020,1007,1006                                            A0011110
 1006 IJK=1                                                              A0011120
      GO TO 998                                                          A0011130
 1100 KK=LIMITU(K)+1                                                     A0011150
      GO TO 1000                                                         A0011160
 1007 II=0                                                               A0011170
      DO 1008 MM=2,JK,1                                                  A0011180
 1008 II=II+JALPH(MM)                                                    A0011190
 1010 IF(IDIAG) 420,420,421                                              A0011191
  421 IKL=1010                                                           A0011198
      PRINT   12, IKL,II                                                 A2011199
  420 IF(II-(JK-2)) 1090,1013,1009                                       A0011200
 1009 IF(II-(JK-1)) 1079,1013,1079                                       A0011201
 1079 IJK=2                                                              A0011210
      GO TO 998                                                          A0011220
 1013 IS(L)=I(J,JJ)                                                      A0011230
      IB(INT,L)=I(J,JJ)
      I1=L
      IPIP=I(J,JJ)
      N10=1
      I2=32
 1836 IF(IPIP-I2) 1833,1834,1835
 1835 IZJ(I1,N10)=1
      IPIP=IPIP-I2
 1833 I2=I2/2
      N10=N10+1
      IF(N10-6) 1836,1836,1831
 1834 IZJ(I1,N10)=1
 1831 CONTINUE
      L=L+1                                                              A0011240
      DO 1012 N=2,JK,1                                                   A0011250
      IF(JALPH(N)) 1014,1012,1015                                        A0011260
 1014 IJK=4                                                              A0011270
      GO TO 998                                                          A0011280
 1015 LIMITU(N)=JSAVE(N)                                                 A0011290
 1012 CONTINUE                                                           A0011300
```

```
 1090 JJ=JJ+1                                                          A0011310
      GO TO 990                                                        A0011320
 1020 K=K+1                                                            A0011330
 1089 IF(JJ-1) 1101,1101,1100                                          A0011340
 1101 KK=LIMITU(K)                                                     A0011350
      GO TO 1000                                                       A0011360
  998 PRINT   997,IJK,(ERROR(IJK,III),III=1,8,1),J,JJ,K,KK,I(J,JJ),    A2012100
     1 L                                                               A2012110
  997 FORMAT(1H ,I2,2X,8A4,70(2X,I3))                                  A0012112
C REMEMBER TO READ IN ERROR MESSAGES                                   A0012113
 2999 PRINT   3015,TAG(1),TAG(2)                                       A2012114
      L=L-1
 3015 FORMAT(1H ,14H FOR THE WORD,2A3)                                 A0012115
      DO 3039 J=1,JK,1                                                 A0012116
      II=ITOTAL(J)                                                     A0012117
 3039 PRINT   12,(I(J,K),K=1,II,1)
      PRINT 2002,(IB(INT,MJ),MJ=1,L,1)
 2002 FORMAT(1H ,33H THE COMMON SEQUENCE OF EVENTS IS,2X,30(I2,1X)/     A0012120
     120(I2,1X))                                                       A0012121
      PRINT 1837, ((IZJ(J1,J2),J2=1,6,1),IB(INT,J1),J1=1,L,1)
 1837 FORMAT(1H ,6I1,2X,I3)
      JTOT(INT)=L                                                      A0012130
      DO 1838 J1=1,25,1
      DO 1838 J2=1,6,1
 1838 IZJ(J1,J2)=0
C INTERCHANGE OF STRINGS TO IMPROVE PREDICTION                         A0012210
      PRINT   12,(IB(INT,K),K=1,L,1)                                    2
      IF(IDIAG) 432,432,433
  433 IKL=880
      PRINT 12,IKL,IWZ,INT                                              2
  432 IF(IWZ-2) 851,3030,3030                                           A0012211
  850 IF(INT-JK) 840,840,860                                            A0012220
  840 N2=N2+1                                                           A0013100
      CALL CHANGE(N2,N3,N4)                                             A0013110
      CALL CHANGE(1,N2,N4)                                              A0013120
      N3=N2                                                             A0013130
      GO TO 3300                                                        A0013139
  851 IF(INT-1) 830,830,850
C FIRST PASS INTERCHANGE
  830 N1=1                                                              A0013200
      N2=2                                                              A0013210
      N3=N2                                                             A0013220
      CALL CHANGE(1,N2,25)                                              A0013230
      GO TO 3300                                                        A0013240
C     CHECK STRINGS                                                     A0014000
  860 PRINT   881                                                       A2014010
  881 FORMAT(1H ,16H COMMON STRINGS FOUND)                              A0014020
      DO 882 J=1,JK,1                                                   A0014030
      PRINT   12,J,(IB(J,K),K=1,N4,1)                                   A2014040
      ITOTAL(J)=JTOT(J)                                                 A0014050
      DO 882 K=1,N4,1                                                   A0014060
  882 I(J,K)=IB(J,K)                                                    A0014070
C SET SWITCH FOR EXIT ON NEXT PASS
      IWZ=2                                                             A0014080
      GO TO 3300
 3030 CONTINUE
C INITIALIZE THE MATRICES                                              A0120010
```

```
      DO 3040 J=1,25,1                                                  A0120020
      IK(J)=0                                                           A0120021
      JTOT(J)=0                                                         A0120022
      IS(J)=0                                                           A0120030
      DO 3040 K=1,25,1                                                  A0120040
      I(J,K)=0
 3040 IB(J,K)=0
      PRINT 69
 3000 CONTINUE                                                          A0123000
 2001 FORMAT(1H1,15H END OF PROGRAM)                                    A0123222
 2000 PRINT   2001                                                      A2123221
      END                                                               A0123223
$     FORTRAN
$      PUNCH OBJECT
C SWITCH ROUTINE                                                        CHANG000
      SUBROUTINE CHANGE(N1,N2,N3)                                       CHANG001
      DIMENSION I(25,25),ITOTAL(25)
      COMMON I
      COMMON ITOTAL
  811 DO 810 K=1,N3,1                                                   CHANG012
      IEMP=I(N1,K)                                                      CHANG013
      I(N1,K)=I(N2,K)                                                   CHANG014
  810 I(N2,K)=IEMP                                                      CHANG015
      IEMP=ITOTAL(N1)
      ITOTAL(N1)=ITOTAL(N2)
      ITOTAL(N2)=IEMP
      RETURN                                                            CHANG016
      END                                                               CHANG017
$      FORTRAN
      SUBROUTINE XCLOR (IORD1,IORD2,IALPH)
      IF (IORD1-IORD2) 1,2,1
    2 IALPH=1
      GO TO 3
    1 IALPH=-1
    3 CONTINUE
      RETURN
      END
```

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*
University of Illinois
Department of Electrical Engineering
Urbana, Illinois 61801

2a. REPORT SECURITY CLASSIFICATION
UNCLASSIFIED

2b. GROUP

3. REPORT TITLE
AUTOMATIC DETERMINATION OF INVARIANCE IN MACHINE CODED SPEECH

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*
Scientific; ; interim

5. AUTHOR(S) *(Last name, first name, initial)*
Schill, John

6. REPORT DATE
December 1, 1968

7a. TOTAL NO. OF PAGES
56

7b. NO. OF REFS
15

8a. CONTRACT OR GRANT NO.
AF-AFOSR 7-67

b. PROJECT AND TASK NO.

c. 9769-04

d. 61445014
681304

9a. ORIGINATOR'S REPORT NUMBER(S)
BCL Report 7.4

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES
Partial sponsorship
NASA NGR 14-005-111 and
AF 33(615)-3890

12. SPONSORING MILITARY ACTIVITY
Directorate of Information Sciences
Air Force Office of Scientific Research
Arlington, Virginia 22209

13. ABSTRACT

A method for determining significant patterns for identification of spoken words was developed. This method is based on the work of Gazdag using his concept of significant subsequences (SSS) which is based upon a machine representation of spoken words. A system was assembled using an algorithm for ascertaining these SSS's implemented on the ILLIAC II. This system consisted of an encoder, a sampler, for the encoder and digital computer. The spoken digits were recognized by this system using the concept of the SSS.

DD FORM 1 JAN 64 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
speech recognition algorithm						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year, or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through

_____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through

_____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through

_____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U).*

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

UNCLASSIFIED

Security Classification