less, for any burst of length 11 or less induces at most a
burst of length 3 or less and one of length 8 or less in the
respective classes of digit positions. Some codes obtained
along the same lines are listed in Table II, where \((b_1, b_2)\)
and \((n_1 + n_2)\) mean that the code is a combination of a
burst-\(b_1\) code with code-length \(n_1\) and a burst-\(b_2\) code
with code-length \(n_2\), each of which is generated by a
polynomial in the first row of Table I.

For any \(b > 20\), it is easy to find a composite shortened
cyclic burst-\(b\) code with the minimum number of check-
digits such that \(n/r\) is nearly equal to 3.

The efficiency of group codes for burst-error correction
might be measured in terms of the theoretical maximum
code length \(n_m\) defined by

\[
n_m = 2^{r+1-k} + b - 2.
\]

Though the composite codes stated above are not so
efficient, \(n/n_m\) for these codes are greater than those for
Fire’s codes with the same \(b\).

ACKNOWLEDGMENT

The author wishes to acknowledge the support and
encouragement of Prof. H. Ozaki, Department of Elec-
tronic Engineering, Osaka University. Also the author
is indebted to S. Matoba of Shin Mitsubishi Heavy
Industries for the programming of IBM-7090.

Tests for Unique Decipherability*

SHIMON EVEN†, MEMBER, IEEE

Summary—A test for unique decipherability of a given code-
word set is described. The test also shows whether the set is de-
cipherable with a finite delay, and in the latter case, determines the
necessary delay. Finally, it is shown how the test of Sardinas and
Patterson [1] can be used to get the same result, as was conjec-
tured by Gilbert and Moore [2].

A TEST FOR UDF

The ideas used in this section are related to Huffman’s
information lossless automata [3, 4].

Let us describe the test on three examples. In the first
example we shall demonstrate how it is detected that a
given code is not UD. The second example is UD but
not UDF. The third is UDF. The alphabet in all our
examples will be \(\Sigma = \{0, 1\}\).

Example 1

Let \(\kappa\) be \(\{0, 0 1, 1 0 1\}\). Let us associate a distinct
symbol (not in \(\Sigma\)) with every one of the words, for instance,

\[
A \ 0
B \ 0 1
C \ 1 0 1 0.
\]

Let us rewrite the code words, inserting intermediate
symbols that will enable us to trace all possible inter-
pretations of a given sequence. If a code word corresponds
to the symbol \(X\) and it consists of \(n (n > 1)\) letters, we
insert the symbol \(X_i\) between its \(i\)th letter, and its
\((i + 1)\)th letter. This is done for all \(1 \leq i < n\). In addition
we write the symbol \(S_i\), standing for “separation” at the
beginning and end of each of the new strings. The resulting
strings in our example are:

\[
S \ 0 \ S
S \ 0 \ B_1 \ 1 \ S
S \ 1 \ C_1 \ 0 \ C_2 \ 1 \ C_3 \ 0 \ S.
\]
Now, let us construct the following table: (See Fig. 1.)

1) The left column is for row headings, the first of which is the symbol $S$. The following columns are headed by the letters of $\mathcal{L}$.

2) If a message starts with an 0, then the message starts either with $A$ or with $B$. The symbol following 0 in $A$ is $S$, and the symbol following 0 in $B$ is $B_1$, therefore, we write in the corresponding entry $(SB_i)$. In this case there are two possibilities, but in general there might be any number of them. If the number of possibilities is one (as it is for the row $S$, column 1—namely, the only possibility is $C_4$) or if there is no such possibility, we leave the entry blank. If the number of possibilities is larger than two, we write in the entry all possible (unordered) pairs. For example, if $a$, $b$ and $c$ are possible symbols, we write in the entry:

$$(ab), (bc), (ac).$$

3) We write all generated compatible pairs in our case just $(SB_1)$ in the left column as heading of new rows and proceed in the following manner: If both symbols are followed by the same letter, we write in the column corresponding to that letter, all new possible compatible pairs; namely, all the possible symbols we might be in at this stage, broken to pairs in the described way. In our example, $B_1$ can be followed only by a 1, so the next symbol there is $C_4$, and $S$ (the second member of the pair), when followed by 1 leads to $C_4$. Therefore, the only pair in this entry is $(SC_4)$. If there are no letters that can follow both symbols in the pair we leave all entries in the corresponding row blank.

4) We repeat 3 for all generated compatible pairs that had not been used as row headings before. Since the number of pairs is bounded, the process must terminate.

5) If at any stage of the table construction we get the pair $(SS)$, the code is not UD. (In our example we get $SS$ for the row $SC_4$.)

As a matter of fact, we can trace back one of the shortest messages that are not uniquely decipherable. (See Fig. 2.) In the right-most place we write the pair with the repeated symbol $(SS)$. Next to the left we write the column heading in which it appeared first, 0 in our example, and next the row heading in which it appeared first. Then we look where this heading $(SC_4)$ appeared first, and proceed in the same manner till we get to the symbol $S$. An ambiguous message is readily determined. In our case it is $0 1 0 1 0$, that can be interpreted both as $AC$ and as $BBA$. (0-1 0 1 0 and 0 1-0 1-0).

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>$(SB_1)$</td>
<td>$(SC_1)$</td>
</tr>
<tr>
<td>$(SB_1)$</td>
<td></td>
<td>$(SC_1)$</td>
</tr>
<tr>
<td>$(SC_1)$</td>
<td>$(SB_2)$, $(B_1C_4)$</td>
<td></td>
</tr>
<tr>
<td>$(SC_2)$</td>
<td></td>
<td>$(C_4C_4)$</td>
</tr>
<tr>
<td>$(B_1C_4)$</td>
<td></td>
<td>$(SC_4)$</td>
</tr>
<tr>
<td>$(C_4C_4)$</td>
<td></td>
<td>$(SB_1)$ $(SS)$</td>
</tr>
</tbody>
</table>

**Fig. 1.**

Example 2

\[ \kappa = \{1, 1 1 0, 0 1 0, 1 0 0\} \]

<table>
<thead>
<tr>
<th>$A$</th>
<th>$S$</th>
<th>$1$</th>
<th>$S$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$B$</td>
<td>$S$</td>
<td>$1$</td>
<td>$B_1$</td>
</tr>
<tr>
<td>$C$</td>
<td>$S$</td>
<td>$0$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>$D$</td>
<td>$S$</td>
<td>$1$</td>
<td>$D_1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td></td>
<td>$(SB_1)$, $(SD_1)$, $(B_1D_1)$</td>
</tr>
<tr>
<td>$(SB_1)$</td>
<td></td>
<td>$(SD_1)$, $(B_1S)$, $(B_1B_1)$, $(B_1D_1)$</td>
</tr>
<tr>
<td>$(B_1S)$</td>
<td></td>
<td>$(B_1D_1)$</td>
</tr>
<tr>
<td>$(B_1B_1)$</td>
<td></td>
<td>$(B_1S)$</td>
</tr>
<tr>
<td>$(B_1D_1)$</td>
<td></td>
<td>$(SD_1)$</td>
</tr>
<tr>
<td>$(C_4D_4)$</td>
<td></td>
<td>$(SC_4)$, $(C_4B_1)$, $(C_4D_1)$</td>
</tr>
<tr>
<td>$(SC_4)$</td>
<td></td>
<td>$(SC_4)$</td>
</tr>
<tr>
<td>$(R_4D_4)$</td>
<td></td>
<td>$(SD_1)$</td>
</tr>
<tr>
<td>$(D_4C_4)$</td>
<td></td>
<td>$(SD_1)$</td>
</tr>
</tbody>
</table>

**Fig. 3.**

Since the pair $(SS)$ was not generated in Fig. 3, the code is UD. Let us now proceed to test whether it is UDF.

Construct a directed graph whose nodes are the row headings of the test table, and whose arcs (labeled with the corresponding letters) are leading to all pairs generated in that row. (See Fig. 4.)

![Fig. 4.](image-url)
The code is UDF if and only if the constructed graph is loop-free. (For an efficient method of determining whether a given graph is loop-free, see Even [4].) In our example the graph is not loop-free, one of the loops being \((SC_1) \rightarrow (SC_2) \rightarrow (SC_1)\). We can construct immediately an infinite message whose first word cannot be uniquely determined. (See Fig. 5.) The constructed message is: 1 1 0 1 0 1 0 · · ·

\[
\begin{array}{c|c}
(S) & (SB_1) \\
\hline
(SB_1) & (SC_1) \\
(sc_2) & (SC_1) \\
(C,C) & \\
\end{array}
\]

The constructed message: 1 1 0 1 0 1 0 · · ·

Now assume that \(\kappa\) is not UD. Then there exists a message with more than one interpretation. Choose two interpretations. These allow two ways of assigning the intermediate symbols, and the corresponding pairs are clearly compatible pairs that will be generated in the table. Among them the final pair is \(\langle SS\rangle\); Q. E. D.

**Theorem 2:** (Analogous to the theorem on page 4 of Even [4].) \(\kappa\) is UDF if and only if it is UD and the derived graph is loop-free.

The proof is analogous to the mentioned theorem, and the construction of infinite message with more than one possible interpretation in case of a loop in the graph is clear from Example 2.

**The Test of Sardinas and Patterson**

In their paper [1], Sardinas and Patterson devised a test for unique decipherability of a given code, however, they were troubled somewhat by the fact that their test sometimes does not terminate. Gilbert and Moore [2] posed the following question: "Are the encodings for which the algorithm of Sardinas and Patterson fail to terminate precisely the same as the encodings having infinite delay?" In the present section we shall show that this is indeed the case, and also show that their test can be used to answer all the questions answered by the previous test. It seems that for some purposes the previous test is more convenient, and for others the test of Sardinas and Patterson is superior.

**Example 3**

\[
\kappa = \{1, 1 0, 0 0 1\}
\]

\[
\begin{array}{c|c}
A & S 1 S \\
B & S 1 B, 0 S \\
C & S 0 C, 0 C_2 1 S \\
\hline
S & 0 1 \\
(SB_1) & (SC_1) \\
(sc_2) & (SC_1) \\
(C,C) & \\
\end{array}
\]

Fig. 6.

Since \(SS\) does not occur in the test table, (Fig. 6) and the graph is loop-free, \(\kappa\) is UDF. \(N(\kappa)\) can be determined from the graph. (See Even [4] for method of determination of a longest path in a loop-free graph) since it is simply one more than the length of a maximal-length path in the corresponding graph. (In our example \(N(\kappa) = 4\).)

The generality of the described procedure is guaranteed by the validity of the following theorems.

**Theorem 1:** A code \(\kappa\) is UD if, and only if, the pair \(\langle SS\rangle\) is not generated in its test table.

**Proof:** Assume that \(\langle SS\rangle\) is generated in the test table. This means that there exists some compatible pair \(ab\) such that the same letter leads \(a\) to \(S\) and \(b\) to \(S\). Since all compatible pairs are reached from \(S\) by two (or more) interpretations of some sequence, the addition of the letter that leads \(ab\) into \(\langle SS\rangle\) will complete the sequence to a message that has two (or more) interpretations, and therefore is not UD.

In Example 1, shown in Fig. 7, \(\operatorname{Seg}_0\) is just \(\kappa\) itself. If \(\kappa\) is prefix [5] then \(\operatorname{Seg}_1\) and all later \(\operatorname{Seg}\)'s are empty. If \(\kappa\) is not prefix, we write in \(\operatorname{Seg}_1\) all the "tails," that is, if \(a\) is a word of \(\operatorname{Seg}_0\) and \(ab\) is a word of \(\operatorname{Seg}_0\) we write \(b\) in \(\operatorname{Seg}_1\). \(\operatorname{Seg}_2\) is constructed in the following way: If \(a\) is a word in \(\operatorname{Seg}_0\) \((\operatorname{Seg}_{-1})\) and \(ab\) is a word in \(\operatorname{Seg}_{-1}\) \((\operatorname{Seg}_0)\), then \(b\) is a word in \(\operatorname{Seg}_1\). (For \(n > 0\) the operation between words of \(\operatorname{Seg}_n\) itself does not take place.) In Example 1 from 0 and 0 1 in \(\operatorname{Seg}_1\) we get 1 in \(\operatorname{Seg}_2\), from 1 in \(\operatorname{Seg}_1\) and 1 0 1 0 in \(\operatorname{Seg}_2\) we get 0 1 0 in \(\operatorname{Seg}_3\). Similarly, 0 1 0 in \(\operatorname{Seg}_2\) and 0 in \(\operatorname{Seg}_0\) give 1 0 1 in \(\operatorname{Seg}_3\), and 0 1 0 in \(\operatorname{Seg}_2\) and 0 1 in \(\operatorname{Seg}_3\) give 0 in \(\operatorname{Seg}_4\). Here we terminate the test since we get a word of \(\operatorname{Seg}_n\) in some other \(\operatorname{Seg}\). (Namely 0 in \(\operatorname{Seg}_1\) is also in \(\operatorname{Seg}_2\).) This means that \(\kappa\) is not UD because by tracing back we can produce a message with at least two interpretations. In our case, 0 is a result of reaction between 0 1 0 \((\operatorname{Seg}_3)\) and 0 1 \((\operatorname{Seg}_0)\); record it this way

\[
\begin{array}{c|c|c|c}
\kappa = \operatorname{Seg}_0 & \operatorname{Seg}_1 & \operatorname{Seg}_2 & \operatorname{Seg}_3 \\
0 & 1 & 0 1 0 & 0 \\
0 1 & 1 0 1 0 & \\
\end{array}
\]

Fig. 7.
The word 0 1 0 is a result of reaction between 1 0 1 0 (Seg0) and 1 (Seg1). Thus,

\[
\begin{array}{c}
0 \\
0 1 0 \\
1 0 1 0 .
\end{array}
\]

The digit 1 is a result of the reaction between 0 (Seg0) and 0 1 (Seg0). Thus,

\[
\begin{array}{c}
0 \\
0 1 0 \\
1 0 1 0 \\
0 1
\end{array}
\]

and we have two interpretations, 0 1 0 1 0 and 0 1 0 1 0, for the message 0 1 0 1 0.

It is now clear that \( x \) is UD if and only if no Seg_\( i > 0 \) contains any word of Seg_\( 0 \).

Let us try the test on Example 2 shown in Fig. 8.

Example 2

<table>
<thead>
<tr>
<th>Seg_0</th>
<th>Seg_1</th>
<th>Seg_2</th>
<th>Seg_3</th>
<th>Seg_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 1 0</td>
<td>1 0</td>
<td>0 0</td>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0 1</td>
<td>1 0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 8.

It is obvious that Seg_2 is the same as Seg_0, and that cycling will occur. Let us show how an infinite message with two interpretations may be constructed.

In the first Seg that is the same as a previous Seg, choose a word that reacts with Seg_0. (It may happen that some do not react with Seg_0 as 0 0 of Seg_0 does not react with Seg_0.) In our example the only word in Seg_4 is 0. Find a word in the previous Seg that generated this word (1 0 in our case) and continue the process until Seg_0 is reached. One such chain in our case may be recorded in the following way.

\[
\begin{array}{c}
0 \\
1 0 \\
0 1 0 \\
1 0 0 \\
1 1 0
\end{array}
\]

The sequence that we get is 1 1 0 0 1 0 \( \ldots \) and it may be interpreted as 1-1 0 0 1-0 1 0 \( \ldots \) or as 1 1 0 0 1 0-1-\( \ldots \) . Since such a message can always be constructed if the process does not terminate, it is clear that nontermination means that the code is of infinite delay. The converse is true too, because if an infinite sequence with two interpretations is given, it is easy to see that the test would not terminate.

Example 3

\[
\begin{array}{c|c|c|c|c|c}
Seg_0 & Seg_1 & Seg_2 & Seg_3 & Seg_4 & empty \\
\hline
1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0
\end{array}
\]

Fig. 9.

Here the test terminated, (Fig. 9) and no word of Seg_0 is a member of any other Seg. Therefore, the code is UDF. Also the order can be found by construction of one of the longest sequences before termination of the test:

\[
\begin{array}{c}
0 1 \\
0 0 1 \\
1 0
\end{array}
\]

We get 1 0 0(1). The sequence 1 0 0 has two interpretations: 1-0 0 (1) and 1 0-0(0 1), but the confusion is ended once the fourth letter is known. Thus, the delay is four.

We have not given precise proofs of our statements about the test of Sardinas and Patterson, because such proofs are tedious and require additional definitions and notation without giving any more insight.

CONCLUSION

We have demonstrated two methods of testing for unique decipherability, both answering all the following questions:

1) Is the code UD? (If not, find an ambiguous finite message.)
2) Is the code UDF? (If not, find an ambiguous infinite message; if so, find the order of its delay.)

It seems that the test of Sardinas and Patterson is more efficient than our test, but if the questions in parenthesis are to be answered, our test is simpler to apply. Also, our test may be extended to more general coding procedures [6].

ACKNOWLEDGMENT

The author is indebted to Dr. M. Cohn, of Sperry Rand Research Center, who criticized the manuscript.

REFERENCES