GÜNEY GÖNENÇ

UNIQUE DECIPHERABILITY OF CODES WITH CONSTRAINTS WITH APPLICATION TO SYLLABIFICATION OF TURKISH WORDS

1. INTRODUCTION

Information lossless automata were first studied by D. A. HUFFMAN (1959). Huffman also devised tests for information losslessness (IL) and information losslessness of finite order (ILF). By treating finite state machines as encoders and decoders, the tests for IL and ILF can be applied to coding theory. This is done by S. EVEN (1962, 1963, 1965) who devised testing methods for unique decipherability (UD) and unique decipherability of finite delay (UDF), concepts shown to be parallel to IL and ILF.

In this paper, tests for UD and UDF for codes with constraints are investigated. The basis of the proposed method is Even's procedure. The constraints are of the form " code word X never follows code word Y " for specific ordered pairs (X, Y) of code words.

The need for testing UD and UDF for codes with constraints originally arised in the syllabification problem for Turkish words. The problem is, essentially, to find an algorithm for syllabification of words for a given printed Turkish text. The construction of syllables in Turkish language is very regular and hence it is not difficult to find such algorithms intuitively, by trial and error. By a thorough analysis of the UD and UDF properties of printed word – syllable structure conversion, it is also possible to investigate the effects of the flood of foreign (mostly French) words on the syllable structure of Turkish.

In part 2 some basic definitions are given. In part 3 Even's procedure for testing UD and UDF is discussed briefly. The test for codes with constraints is presented in part 4. Finally, in part 5, applications on Turkish syllable structure are discussed briefly.

2. NOTATION AND BASIC DEFINITIONS[1]

Source symbols will be shown by capital letters A, B, ..., L, W, X, Y, Z. Code symbols will be shown by 0 and 1. A concatenation of a finite number of code symbols is called a *code word.* A *code* consists of a finite number of code words of finite length, each representing a source symbol. A *coded message* is obtained by concatenating code words, without spacing or any other punctuation. *Variable-length codes* in which code words are not necessarily of the same length, will only be considered.

A code is said to be *uniquely decipherable* if and only if every coded message can be decomposed into a sequence of code words in only one way. A code is said to be *uniquely decipherable of finite delay* N if and only if N is the least integer, so that the knowledge of the first N symbols of the coded message suffices to determine its first code word.

3. TESTS FOR UD AND UDF

By treating finite state machines as encoders and decoders, tests for UD and UDF can be converted into tests for IL and ILF (S. EVEN, 1965; Z. KOHAVI, 1970). Without going into tests for IL and ILF, we shall give Even's testing procedure for UD and UDF here. At the same time we shall demonstrate the procedure on a binary code T which consists of 4 code words:

$$A = 0, \quad B = 10, \quad C = 01, \quad \text{and} \quad D = 101.$$[2]

Procedure 1.

(1*a*) Insert a separation symbol S at the beginning and end of each code word in the code.

(1*b*) Let code word X be of length n. Insert the separation symbol X_i between i-th and $(i+1)$-th symbol of code word X for $1 \leqslant i \leqslant n$-1. Do

---

[1] Basic definitions in parts 2 and 3 follows Z. KOHAVI (1970).

[2] If 0 denotes vocal and 1 denotes consonant, then A, B, C, and D are four of the 12 syllable types of Turkish.

this for all code words for which $n \geqslant 2$. For example, after steps (1*a*) and (1*b*), $D = 101$ becomes $D = S1D_1 0D_2 1S$.

(1*c*) The separation symbol to the right of the code symbol t is called the *t-successor* of the separation symbol to the left of the same code symbol. For example, D_1 is the 1-successor of S, D_2 is the 0-successor of D_1, and S is the 1-successor of D_2, in code word D.

Two separation symbols are said to be *compatible* if

(I) They are t-successors of S, for some code symbol t, or,

(II) They are t-successors of two separation symbols which are themselves compatibles.

If (WX) is a compatible pair, and if Y and Z are t-successors of W and X, respectively, then the compatible pair (YZ) is said to be *implied* by (WX) under t.

Construct a testing table as follows: the column headings are the code symbols. The first row heading is S. The entries in the first row are compatible pairs found by (I) above, under corresponding column t. The other row headings are the compatible pairs. The entries in row (WX), column t, are the compatible pairs implied by (WX) under t. The testing table for code T is shown in fig. 1.

code T		0	1
A: 0	S	(SC_1)	(B_1D_1)
B: 10	SC_1	—	(SB_1) (SD_1)
C: 01	B_1D_1	(SD_2)	—
D: 101	SB_1	(SC_1) (SS)	—
A: $S0S$	SD_1	(C_1D_2) (SD_2)	—
B: $S1B_10S$	SD_2	—	(SB_1) (SD_1)
C: $S0C_11S$	C_1D_2	—	(SS)
D: $S1D_10D_21S$			

Fig. 1. *Code T*

(1*d*) If the table contains pair (SS) then the code is not UD, otherwise it is UD. Since there are (SS) pairs in the testing table for code T, it is not UD. By tracing back the compatibles, starting from a (SS) pair, one can arrive the symbol S (possibly through several paths).

The sequence of code symbols corresponding to this traceback path gives an ambiguous message. In fig. 2 some of these ambiguous messages are shown for code T.

$$S \overset{0}{—} SC_1 \overset{1}{—} SB_1 \overset{0}{—} SS \qquad 010 = AB \text{ or } CA$$

$$S \overset{0}{—} SC_1 \overset{1}{—} SD_1 \overset{0}{—} C_1D_2 \overset{1}{—} SS \qquad 0101 = AD \text{ or } CC$$

$$S \overset{1}{—} B_1D_1 \overset{0}{—} SD_2 \overset{1}{—} SB_1 \overset{0}{—} SS \qquad 1010 = BB \text{ or } DA$$

$$S \overset{1}{—} B_1D_1 \overset{0}{—} SD_2 \overset{1}{—} SD_1 \overset{0}{—} C_1D_2 \overset{1}{—} SS \qquad 10101 = BD \text{ or } DC$$

Fig. 2. *Some ambiguous messages in code T*

(1*e*) If no (*SS*) pair is generated, then a testing graph is constructed from the table as follows: corresponding to every row in the table there is a vertex in the graph. If (*YZ*) is implied by (*WX*) under *t*, then a directed arc labeled *t* leads from vertex (*WX*) to vertex (*YZ*) in the graph.

(1*f*) A code is uniquely decipherable of finite delay *N* if and only if its testing graph is loop-free. If the graph is loop-free and the length of the longest path in the graph is *r*, then $N = r + 1$.

4. CONSTRAINTS ON CODE WORD OCCURRENCES

In the above discussion, there was no constraint whatsoever regarding the occurrence of any code word at any point of the message. On the other hand there may be such a case that, for some specific code, the code word *X* never follows the code word *Y*. These constraints may arise from the physical nature of the encoder (for example no letter other than *u* can follow letter *q* in an English text) or may be deliberately imposed upon a code to achieve UD or UDF properties.

The constraints of the form " code word *X* never follows code word *Y*" will be termed a *first-order* constraint. For the codes with first order constraints, a testing procedure is given below:

Procedure 2.

(2*a*) Insert a separation symbol P_X at the beginning and a separation symbol Q_X at the end of each code word *X* in the code.

(2*b*) Insert separation symbols X_i as in (1*b*). For example after steps (2*a*) and (2*b*), $D = 101$ becomes $P_D1D_10D_21Q_D$.

(2*c*) Let a number $m(X, Y)$ be defined for every ordered pair of code words (*X*, *Y*) in the following way:

$m(X, Y) = 1$ if the code word X is allowed to occur immediately after the code word Y,
$= 0$ otherwise.

A *constraint matrix* M in which there is one row and one column for each code word can be defined such that the element of M in the row X, column Y is $m(X, Y)$.
For example, consider code T of part 3. Let the following four constraints be imposed on this code: A never follows C,. C never follows C, A never follows D, and C never follows D. These four constraints can also be expressed as " a code word starting with a 0 never follows a code word ending with a 1 ". The resulting code, called code U, and its constraint matrix is shown in fig. 3.

code U
A: 0
B: 10
C: 01
D: 101
A: $P_A 0 Q_A$
B: $P_B 1 B_1 0 Q_B$
C: $P_C 0 C_1 1 Q_C$
D: $P_D 1 D_1 0 D_2 1 Q_D$

	0	1
P	$(Q_A C_1)$	$(B_1 D_1)$
$Q_A C_1$	—	$(Q_C B_1)$ $(Q_C D_1)$
$B_1 D_1$	$(Q_B D_2)$	—
$Q_C B_1$	—	—
$Q_C D_1$	—	—
$Q_B D_2$	—	$(Q_D B_1)$ $(Q_D D_1)$
$Q_D B_1$	—	—
$Q_D D_1$	—	—

$$M = \begin{array}{c} \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Fig. 3. *Code U*

(2*d*) The separation symbol to the right of the code symbol t is called the *t-successor* of the separation symbol to the left of the same code symbol. Furthermore, a separation symbol X_i (Q_X) is the t-successor of the separation symbol Q_Y if X_i (Q_X) is a t-successor of P_X and $m(X, Y) = 1$. Two separation symbols are said to be *compatible* if
(I) They are t-successors of P_X and P_Y for some t, X, and Y, or

(II) They are t-successors of two separation symbols which are themselves compatible.

Construct the testing table as in (1c), with the change: the first row heading is P. The testing table for code U is shown in fig. 3.

(2e) If the table contains any pair $(Q_X Q_Y)$ for some X and Y (possibly identical), then the code is not UD. Otherwise it is UD. For example it is seen from fig. 3 that code U is UD. If the code is not UD, then a traceback of compatibles which implied a pair $(Q_X Q_Y)$ gives an ambiguous message.

(2e) If the code is UD, then one can construct the testing graph as in (1e). The testing graph for code U is shown in fig. 4.

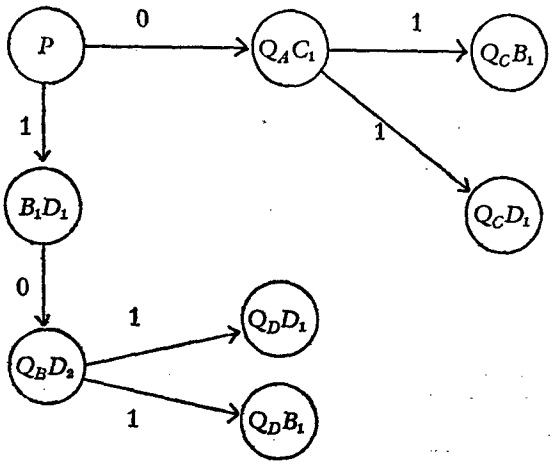

Fig. 4. *Testing graph for code U*

The longest path in this graph has length 3. Hence the code is UDF of order 4; in other words the knowledge of the first 4 code symbols suffices to determine the first code word, but 3 is not sufficient. To demonstrate that the knowledge of the first 3 code symbols is not sufficient, consider a path of length 3 in the graph, for example the path 101 from P to $Q_D D_1$. When we receive 101 we can not decide whether this is word D, or word B (= 10) occurred and a word D (= 101) has just started (the last vertex $Q_D D_1$ actually points to this ambiguity). But, if the fourth symbol received is a 0 we can now decide that the first code word was B, and if the fourth symbol is a 1 we decide that the first code word was D.

There may be other types of constraints present on the code. A constraint of the form " code word X never follows YZ ", where Y and

Z are distinct, will be termed a *second order* constraint. If there exists such a constraint, then it can be converted into the following first order constraints: create a new code word $\bar{Z}$, identical in structure to Z. Then impose the constraints "X never follows $\bar{Z}$, Z never follows Y" (for simplification purposes one can impose the additional constraints: "$\bar{Z}$ never follows $\bar{Z}$, X, or Z"). Higher order constraints can be handled similarly.

5. SYLLABLE STRUCTURE OF TURKISH LANGUAGE

In Turkish language there are 12 syllable types. These are shown in Table 1.

TABLE 1.

SYLLABLE TYPES OF TURKISH LANGUAGE
(0 denotes vowel, 1 denotes consonant)

Symbol	*Structure*	*Example*
A	0	*a*çık (open)
B	10	*ba*ba (father)
C	01	*ek*mek (bread)
D	101	al*tın* (gold)
E	011	*erk* (power)
F	1011	*türk* (turkish)
G	110	*kra*liçe (queen)
H	1101	kon*trol* (control)
I	1110	*stra*teji (strategy)
J	11101	*stron*siyum (strontium)
K	11011	*trenç*kot (trench coat)
L	10111	*kontr*bas (cello)

The first six syllable types (types *A*-*F*) are syllable types of proper Turkish language. The remaining six types (types *G*-*L*) came into Turkish with foreign borrowings. These are somewhat characterized by consonant clusters, which are totally alien to the language. In spoken language, especially as spoken by not-well-educated people, these clusters are simplified by the addition of a vowel before or within the

cluster, thereby increasing the number of syllables in the word (G. L. LEWIS, 1967). Since our main concern is printed texts we shall not deal with these and other aspects of the spoken language.

The treatment of printed Turkish words as messages encoded into a code in which syllables are code words and letters are code symbols enables us to syllabify printed texts automatically. This is important because of the following reasons:

1) Automatic syllabification makes it possible to recognize and count (mainly for statistical purposes) syllable types and/or syllables from texts read into the computer without any syllable separation markers.

2) Automatic syllabification is necessary in automatic typesetting, without automatic syllabification words to be separated at line ends can not be properly syllabified.

3) Automatic syllabification gives insight into the syllable structure, its deformation under some effects, and the relation between spoken and printed language, thereby helping linguists working on the subject.

The first six syllable types[3] without any constraints obviously form a non-UD code. For example a word 0110 can be decoded as 01.10 (*CB*) or as 011.0 (*EA*). On the other hand the phonetic rules of the language put some constraints as to which syllable type can not follow a given syllable type. The set of constraints inherent in the language can be summarized as "each vowel takes the first consonant before it into its syllable" (T. BANGUOĞLU, 1959). In our notation, the constraint set can be summarized as "no syllable starting with a vowel (0) can follow a syllable ending with a consonant (1)". The constraint matrix corresponding to this set is shown below.

	A	*B*	*C*	*D*	*E*	*F*
A	1	1	0	0	0	0
B	1	1	1	1	1	1
C	1	1	0	0	0	0
D	1	1	1	1	1	1
E	1	1	0	0	0	0
F	1	1	1	1	1	1

[3] Turkish alphabet consists of eight vowels (*a, e, ı, i, o, ö, u, ü*) and 21 consonants (*b, c, ç, d, f, g, ğ, h, j, k, l, m, n, p, r, s, ş, t, v, y, z*). Only one vowel can be present in any syllable. There are no diphtongs in Turkish.

Now, by constructing the testing table and graph, it can be shown that this code is UDF of order 5.[4] This simply means that there is an algorithm, to syllabify any proper Turkish word which operates in the following manner:

1) The only information required about the characters in the text is about their being vowel, consonant or "other" (such as blank, comma, numeral etc.).

2) When a word is being scanned, its first syllable will be decided upon at the fifth character of the word or before. Since the code is UD the decision process is completed when the word ends (i.e. upon first blank).

The introduction of the syllable types $G, H, \ldots, L$ of Table 1 into the language causes the "invention" of new constraints. These are not yet thoroughly investigated or explained. One set of constraints can be summarized as: "no syllable starting with two or more consonants can follow a syllable ending with a vowel".[5] With the addition of this set of contraints, the constraint matrix becomes

	A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	*I*	*J*	*K*	*L*
A	1	1	0	0	0	0	1	0	1	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1	1
C	1	1	0	0	0	0	1	0	1	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1
E	1	1	0	0	0	0	1	0	1	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1	1
G	0	0	1	1	1	1	0	1	0	1	1	1
H	0	0	1	1	1	1	0	1	0	1	1	1
I	0	0	1	1	1	1	0	1	0	1	1	1
J	0	0	1	1	1	1	0	1	0	1	1	1
K	0	0	1	1	1	1	0	1	0	1	1	1
L	1	1	1	1	1	1	1	1	1	1	1	1

[4] It is also interesting to note that the first order constraints to make the code $A, B, \ldots, F$ uniquely decipherable of finite delay are found to be *precisely* those constraints inherent in the language.

[5] No mention of this kind of constraint is found in the literature. This rule, and the one given before must clearly be the result of the shape of vocal organs. We should also mention that *no exception at all* to these two rules exists.

The code thus generated can be shown to be still non-UD. Some typical ambiguities concerning the existing words are shown below:

Message (word)	*Ambiguous syllabification*		*Examples*	
01110	*CG*	01.110	*em.pri*.me,	*is.pri*.tiz.ma
	EB	011.10	*ens.ti*.tü,	*esk.ka*.va.tör
101110	*DG*	101.110	*kon.gre*,	*kom.pra*.dor
	FB	1011.10	*fark.lı*,	*kürk.çü*
011101	*CH*	01.1101	*ıs.tran*.ca,	*ak.tris*
	ED	011.101	*eks.per*	
1011101	*DH*	101.1101	*kan.gren*,	*kon.trat*
	FD	1011.101	*tabl.dot*,	*teks.til*

A careful and thorough search (through all borrowings in the language) revealed one fact: if we increase the code symbols from two (vowel, consonant) to three (v = vowel, r = letter "r", $\bar{r}$ = consonant other than "r") then the resulting code becomes UD, and actually UDF of delay 7 for all existing foreign (and of course all native) words. The examples given above hints this. Simply note that the words in the upper line in each set have an r as the second letter of second syllable, whereas a letter other than r appears at the same position of the word, for words of the lower lines, e.g. *emprime* and *enistitü*.

Finally, with these considerations an algorithm for the syllabification is programmed (in FORTRAN). This algorithm is based on the state-table of the inverse of the finite state machine which is taken as the encoder device 4,7. The input to the program is a printed text, the output is the same text (numerals etc. skipped), all the words being syllabified. There are minor additions to the program. For example unsyllabifiable words (due to punching errors, etc.) are printed out as they are, but in brackets. The program is run on IBM 360/40. An example of input data and corresponding printouts are shown in fig. 5.

HECE AYIRMA PROGRAMI GELENEK AKARYAKIT UYGULAMA
HE*CE A*YIR*MA PROG*RA*MI GE*LE*NEK A*KAR*YA*KIT UY*GU*LA*MA

TORTU KONGRE KORKAK KANGREN TABLDOT KONTRAT TANJANT
TOR*TU KON*GRE KOR*KAK KAN*GREN TABL*DOT KON*TRAT TAN*JANT

STEREOSKOP AHMET RIZA O STRC BB ANI
STE*RE*OS*KOP AH*MET RI*ZA O (STRC) (BB) A*NI

EMPRIME ENSTITU EKSPER ISTRANCA ISTRONGILOS ISFENKS
EM*PRI*ME ENS*TI*TU EKS*PER IS*TRAN*CA IS*TRON*GI*LOS IS*FENKS

FBRKET (KANDIRMACA) 12/MAYIS/1971 GUSULHANE
(FBRKET) KAN*DIR*MA*CA MA*YIS GU*SUL*HA*NE

SAAT TATAR AMFITEATR TELEKS KREOZOT FLAMA FLUOR
SA*AT TA*TAR AM*FI*TE*ATR TE*LEKS KRE*O*ZOT FLA*MA FLU*OR

AERODINAMIK AIT ARAP AORT AVURT ARKEOLOG BABA
A*E*RO*DI*NA*MIK A*IT A*RAP A*ORT A*VURT AR*KE*O*LOG BA*BA

TRAHOM FREKANS STRATEJI STRATOSFER ARTI
TRA*HOM FRE*KANS STRA*TE*JI STRA*TOS*FER AR*TI

KONTRAST EKSKAVATOR ENSTITU
KON*TRAST EKS*KA*VA*TOR ENS*TI*TU

Fig. 5. *Computer printouts of the syllabification program. In each set of two lines, the upper line is the input data, the lower line is the output.*

REFERENCES

T. BANGUOĞLU, *Türk Grameri-Ses Bilgisi*, Ankara, 1959.

S. EVEN, *Generalized Automata and Their Information Losslessness*, in *Switching Circuit Theory and Logical Design*, AIEE Special Publication, S-141, 1962, pp. 144-147.

S. EVEN, *Tests for Unique Decipherability*, in « IEEE Trans. Information Theory », vol. IT-9 (April 1963), pp. 109-112.

S. EVEN, *On Information Lossless Automata of Finite Order*, in « IEEE Trans. Elec. Comp. », vol. EC-14 (August 1965), pp. 561-569.

D. A. HUFFMAN, *Canonical Forms for Information Lossless Finite State Machines*, in « IRE Trans. Circuit Theory », vol. CT-6, Special Supplement (May 1959), pp. 41-59.

G. L. LEWIS, *Turkish Grammar*, London, 1967.

Z. KOHAVI, *Switching and Finite Automata Theory*, New York, 1970.