J. Univ. Kuwait (Sci.) **11**, 1984

A mathematical model for an Arabic word generator

F. A. MUSA

Department of Mathematics, University of Kuwait

ABSTRACT

In Arabic script as well as in the script of other similar languages the shape of a letter changes from one word to another. The number of different shapes for some letters may be three or even four depending on the neighbouring letters in the word. For this reason it became usual to allocate more than one key for the same letter in the Arabic typewriter and computer terminal keyboards. In recent years and with the introduction of word processors, the Arabic keyboard was modified and the number of keys representing the same letter was reduced as a result of applying contextual analysis to each character in order to choose the correct shape of the character. Most of the context analyses used lack generality and are hardly satisfactory. In this paper a mathematical model for an Arabic script generator is introduced and the model is reduced to an algorithm which was implemented on a microcomputer to generate Arabic script using one key only for each character.

INTRODUCTION

The basic alphabet of the Arabic language consists of a certain number of letters and a few long and short vowels and diacritics (Table 1). Some of the Arabic letters have one shape only, others have two or three or even four shapes. The shape of each letter depends on the neighbouring letters in the string of letters of an Arabic word. The pronunciation of any Arabic letter does not depend on its shape but, as in most natural languages, on the meaning of the word. The vowels and diacritics are used in Arabic to indicate the proper pronunciation of a letter and are omitted in many of the recent Arabic publications.

The problem of representing the Arabic characters was considered early in this century when typewriters were introduced and was solved by allocating a key for each letter shape or fount; the result of this solution was slow typing. When computers were introduced into the Arab world, computer terminal keyboards looked exactly the same as typewriter keyboards.

In recent years many researchers in the Arab world (e.g. Al-Banna 1980) and many software developing firms outside the Arabic world started to look into the problem of representing Arabic characters. Most of the researchers in this field have classified Arabic letter shapes or founts into 'first position fount', 'middle position fount' and 'last position fount' (Al-Banna 1980). This classification is inadequate since some

Table 1. Extended Arabic character set

Reference no. R	Proposed corresponding Latin ascii	First shape S_1	Second shape S_2	Third shape S_3	Fourth shape S_4
0		null			
Set R_2					
1	89	ء	ع	ـع	ـع
2	84	غ	غ	ـغ	ـغ
3	68	ي	ي	ـي	ـي
4	66	ى	ى	ـى	ـى
5	85	ه	ه	ـه	ـه
Set V					
6	39	ـَ (فتحة)			
7	60	ـِ (كسرة)			
8	38	ـُ (ضمة)			
9	35	ـّ (شدة)			
10	34	ـً (فتحتان)			
11	62	ـٍ (كسرتان)			
12	91	ـٰ (مدة)			
13	01	ـْ (سكون)			
14	02	ـٌ (ضمتان)			
15	18	ـٔ (همزة وصل)			
Set D					
16	32	فراغ			
17	33	!			
18	61	=			
19	58	,			
20	15	،			
21	37	÷			
22	05	،			
23	04	؛			
24	40	(			
25	41	)			
26	42	×			
27	43	+			
28	36	؟			
29	45	−			
30	31	.			
31	64	:			
32	48	٠			
33	49	١			
34	50	٢			
35	51	٣			

Table 1 (*cont.*)

Reference no. R	Proposed corresponding Latin ascii	First shape S_1	Second shape S_2	Third shape S_3	Fourth shape S_4
36	52	٤			
37	53	٥			
38	54	٦			
39	55	٧			
40	56	٨			
41	57	٩			
Set IND					
42	72	ا			
43	86	د			
44	67	ذ			
45	77	ر			
46	46	ز			
47	44	و			
48	78	ة			
Set R_1					
49	70	بـ	ب		
50	74	تـ	ت		
51	87	ثـ	ث		
52	80	جـ	ج		
53	79	حـ	ح		
54	73	خـ	خ		
55	83	سـ	س		
56	65	شـ	ش		
57	81	صـ	ص		
58	47	ضـ	ض		
59	88	طـ	ط		
60	90	ظـ	ظ		
61	82	فـ	ف		
62	69	قـ	ق		
63	59	كـ	ك		
64	71	لـ	ل		
65	76	مـ	م		
66	75	نـ	ن		
67	63	ئـ	ئ		

letters have more than one fount in the middle or last position, e.g. Character 1 in Table 1 may have Shape S_2 or Shape S_4 in the last position.

In this work we address the above problem by building a mathematical model that reflects accurately the characteristics of Arabic strings and we introduce a model for an Arabic script generator. Thus we introduce a formal system and the rules for

generating this system. This technique was followed by Hyder (1972) in describing a system for generating Urdu script. The rules we shall introduce are much fewer and simpler than those suggested by Hyder. Since our system is programmable, we have implemented the system on a microcomputer to verify that it works and we have found that the need for more than one key for the same letter was eliminated.

The program is rather simple and may be implemented on any microcomputer. We have implemented it on the Apple II Plus microcomputer after we managed to write a program to store and soft generate all the Arabic characters under consideration. The program and its output are given in Figs 1 and 2.

MATHEMATICAL PRELIMINARIES

Let R and S be finite sets with cardinalities n and m respectively, where R is a subset of the set of integers I and S the set of all letter founts in any natural language, then we may introduce the following definition.

Definition 1. A function C is called a character generator if it maps every element $x \in R$ to a unique element $y \in S$ such that $C(x) = y$.

Let R^* denote all integer vectors with elements from R, and S^* be the space of all strings of finite length defined over S, then we may have the following definition.

Definition 2. A function G is called a script generator if for any vector $\mathbf{X} \in R^*$ $\exists$ a string $Y \in S^*$ such that $G(\mathbf{X}) = Y$.

In many natural languages, e.g. English, C is usually a bijection $C: R \rightarrow S$ and thus the construction of a script generator G becomes an easy task and we may state this in the following theorem.

Theorem 1. If C is a bijection then

$$G(\mathbf{X}) = \prod_{x_i \in \mathbf{X}} C(x_i)$$

where

$$\prod_{x_i \in \mathbf{X}} C(x_i) = C(x_0)\, C(x_1) \ldots C(x_n)$$

where $\mathbf{X}$ is a vector of order $n+1$.

In the Arabic language C is not a bijection and this is why the construction of an Arabic script generator is hard. For this reason we shall consider in the following some of the techniques that may be used to replace C by a set of bijection functions. Let us first consider the following theorem which indicates that this is possible.

Theorem 2. Let R and S be finite sets with cardinalities n and m respectively, then there are m^n functions from R to S.

Proof. Stanat & McAllister (1971, p. 221).

Corollary 1. For finite sets R and S, $\exists\ 2^{|R|}$, i.e. 2^n; distinct subsets of R and $2^{|S|}$ distinct subsets of S.

Proof. Stanat & McAllister (1971, p. 221).

Applying the previous properties and definitions to the Arabic language we may construct the mathematical model starting with the following: let $R=\{0, 1, \ldots, 67\}$ be the set of reference numbers given in Table 1, and let

$$\begin{aligned} R_1 &= \{49, \ldots, 67\} \\ R_2 &= \{1, \ldots, 5\} \\ V &= \{6, \ldots, 15\} \\ D &= \{16, \ldots, 41\} \\ \text{IND} &= \{42, \ldots, 48\} \end{aligned}$$

be proper subsets of R, where R_1 is the set of reference numbers of characters that have two shapes, R_2 of characters that have four shapes, V the diacritics and IND for characters that have only one shape and may link to other character shapes at their right side only. D is the set of delimiters as given in Table 1.

Since each character has at least one shape we may group all those shapes in one set called the S_1 set of shapes, and S_2 is the set of second shapes for characters having more than one shape and S_3 and S_4 are the sets of third and fourth shapes. The sets $S_1 \ldots S_4$ appear in Table 1.

Moreover let

$$\begin{aligned} &F_1: R \to S_1 \\ &F_2: R_1 \cup R_2 \to S_2 \\ &F_3: R_2 \to S_3 \\ &F_4: R_2 \to S_4 \end{aligned}$$

be bijections defined on the given domains and codomains. Then we may proceed to establish the mathematical model for a script generator. Noticing first that the omission or inclusion of diacritics has no effect on the process of letter shape selection in Arabic, we may reflect this fact in the following definition.

Definition 3. Two Arabic script strings W_1 and W_2 are said to be equivalent if and only if

$$W_1 \subseteq W_2$$

and

$$(W_2 - W_1) \subset V \quad .$$

Thus in view of Definition 3 the following model for an Arabic script generator will analyse the equivalent consonant script of the one that contains diacritics. Nevertheless we do not neglect diacritics from our contextual analysis and this is included implicitly in our mathematical model and explicitly in the following algorithm.

THE MODEL

Let $\mathbf{W}$ be any vector in the form

$$\mathbf{W} = \begin{pmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \end{pmatrix}$$

where each **x**, **y**, **z** is a vector from the set R^* defined over R as given in the previous section. Then G is called 'Arabic script generator' if

$$G(\mathbf{W}) = G(\mathbf{x})\, G(\mathbf{y})\, G(\mathbf{z})$$

where for any

$$\mathbf{x} = (x_{i-1}, x_i, x_{i+1})^T$$

we have

$$G(\mathbf{x}) = C(x_{i-1}/x_{i-2}, x_i)\, C(x_i/x_{i-1}, x_{i+1})\, C(x_{i+1}/x_i, x_{i+2})$$

where x_{i-2} is the element before x_{i-1} and x_{i+2} is the element after x_{i+1}, and for any vector $(a, b, c)^T$ we have

$$C(b/a, c) = F_2(b) \text{ for } \{a \in R, b \in R_1, c \in D\} \text{ or } \{a \in (\{0\} \cup D \cup \text{IND}), b \in R_2, c \in D\} \quad \text{(I)}$$

$$= F_3(b) \text{ for } \{a \in (R_1 \cup R_2), b \in R_2, c \notin D\} \quad \text{(II)}$$

$$= F_4(b) \text{ for } \{a \in (R_1 \cup R_2), b \in R_2, c \in D\} \quad \text{(III)}$$

$$= F_1(b) \text{ otherwise.} \quad \text{(IV)}$$

PROCEDURE

Starting with an input vector of reference numbers which represents an Arabic script, i.e. word, sentence, paragraph, we scan the vector applying the previous rules and outputting the proper shapes starting at the right and concatenating to the left as with the Arabic script. We obtain the required script.

Example. Consider the vector

$$\mathbf{W} = (0, 42, 64, 05, 47, 42, 52, 01, 16, 0)^T$$

Step no.	Vector	Output	Rule no.
	0		
	42		
1	64	ا	IV
2	05	الـ	IV
3	47	الهـ	II
4	42	الهـو	IV
5	52	الهـوا	IV
6	01	الهـواج	IV
7	16	الهـواجع	III
8	0	space الهـواجع	IV

The generator **G** may be constructed through the following algorithm, since the algorithm uses the succeeding as well as the preceding characters in order to determine the exact shape of the current character. A temporary character shape is echoed which is later modified when the algorithm can decide the exact shape.

THE G ALGORITHM

```
        Flag ← 1            ;  Flag = 1 indicates that the a character is
                               ∈{∅} ∪ IND ∪ V ∪ D
           $ ← " "          ;  indicates current character b ∈ {∅} ∪ IND ∪ V ∪ D
L∅ :    Get b
L1 :    IF b = "END" Then stop; ENDIF
        IF b ∈ IND ∪ V
          Then      $ ← " "
                    Flag ← 1
                    Go to L3
        ENDIF ;
        IF      b ∈ R1
          Then      $ ← "2"
                    Flag ← ∅
                    Go to L3
        ENDIF ;
        If b ∈ R2
          Then   $ ← "4"
                 IF Flag = ∅ Then Draw F3(b) Go to L4
                 ENDIF
        ENDIF ;
L3 :    Draw F1 (b)
        IF      $ = " " Then Go to L∅ ENDIF ;
L4 :    Get c
          IF  c ∉ V ∪ D
             Then  IF $ ≠ " " Then Flag ∈ ∅ ENDIF
                   b ← c
                   Go to L1
        ENDIF
        IF c ∈ V Then  Draw F1 (c)
                       Go to L4
        ENDIF ;
        IF  c ∈ D and ( $ = "2" or ( $ = "4" and Flag = 1))
            Then  Replace F1 (b) by F2(b)
                  Go to L5
        ENDIF ;
        Replace F3(b) by F4(b)
L5 :      $ ← " "
        Draw F1 (c)
        Flag ← 1
        Go to L∅
```

FIG.1.SOURCE LISTINGS FOR FLINE-EDITOR

```
10  LOMEM: 18432
20  REM  ********************************
30  REM         FLINE-EDITOR
40  REM       F. MUSA AND K. GHAWI
50  REM           1983
60  REM  REFERENCE: A MATHEMATICAL MODEL FOR GENERATING ARAB
IC SCRIPT
70  REM  ********************************
80 I1 = 94:I3 = 105:X0 = 272:Y0 = 14:AINDEX = 1:B = 67:SP =
23:MS = 9:MR = 175:DD = 105:FLAG = 1:SET$ = ""
90 PO% = 0:BB% = 0:QB% = 0:LL% = 0:N2 = 69
100  DIM RX(I1),F1%(I1),SZ%(I3),CD%(N2),OLO%(N2),INDP%(N2):I
NDP%(0) = 1
110  FOR J = 1 TO N2:OLO%(J) = 83:CD%(J) =  - 1: NEXT J: REM
  INITIALIZE ARRAYS
120  REM  ==LOADING THE NEW SHAPE TABLE==
130 D$ =  CHR$ (4): PRINT D$;"BLOAD BINSHAPE,A16384"
140  POKE 232,00: POKE 233,64: REM  S=16384
150  REM  ==LOADING THE MAPPING FUNCTION==
160  PRINT D$;"OPEN M1": PRINT D$;"READ M1": FOR J = 1 TO I3
: INPUT SZ%(J): NEXT : FOR J = 0 TO I1: INPUT F1%(J): INPUT
RX(J): NEXT
170  PRINT D$;"CLOSE"
180  REM  ********* MAIN PROGRAM **********
190  SCALE= 1: HGR : ROT= 0: POKE  - 16302,0: HCOLOR= 3:Y =
Y0
200 X = X0: GOSUB 230: GOSUB 300: IF RB = 27 THEN  END
210 AINDEX = 1:Y = Y + SP:FLAG = 1: GOTO 200
220  REM  ------------------------------
230  REM === SUB FOR DRAWING SAHEM ===
240  IF Y > MR THEN  HGR : POKE  - 16302,0:X = X0:Y = Y0
250  DRAW DD AT X,Y
260  RETURN
270  REM === SUB FOR REMOVING SAHEM ===
280  XDRAW DD AT X,Y
290  RETURN
300  REM  =====SUB FOR LINE-EDITOR========
310  GET B$:ACKB =  ASC (B$):RB = RX(ACKB): IF (RB <  > 21)
AND (RB <  > 27) AND (RB <  > 13) THEN BB% = 1:CD%(AINDEX) =
 RB: IF  NOT (PO% = 1) THEN BB% = 0: GOTO 520
320  IF BB% = 1 THEN BB% = 0: GOTO 340
330  GOTO 420
340  IF SZ%(SH) = 0 THEN BINDEX = BINDEX - 1:FLAG = INDP%(BI
NDEX - 1):SH = OLO%(BINDEX):ADR = SH: IF ADR <  >  - 1 THEN
340
350 PO% = 0: IF INDP%(BINDEX - 1) = 2 THEN FLAG = INDP%(BIND
EX - 2): IF FLAG = 2 THEN FLAG = INDP%(BINDEX - 3)
360  IF ADR = B THEN FLAG = 1: GOTO 520
370 RC = RB
380 RB = CD%(BINDEX): IF RB >  = 8 AND RB <  = 18 THEN BINDE
X = BINDEX - 1:FLAG = INDP%(BINDEX - 1): IF FLAG = 2 THEN FL
AG = INDP%(BINDEX - 2): IF FLAG = 2 THEN FLAG = INDP%(BINDEX
 - 3)
390  IF RB >  = 8 AND RB <  = 18 THEN 380
400 ADR = OLO%(BINDEX):SH = ADR: IF RB >  = 19 AND RB <  = 7
1 THEN 610
410 QB% = 1:T = AINDEX:AINDEX = BINDEX: GOSUB 270: GOSUB 770
: GOSUB 230: GOTO 520: REM    PRESS *CTRL->*TO CORRECT MISTA
KES,*ESC*TO END THE PROGRAM
```

```
411  REM
412  REM
413  REM
414  REM
415  REM
420  IF RB = 21 THEN PO% = 1:SH = ADR: GOSUB 270: GOSUB 770:
 GOSUB 230:AINDEX = AINDEX - 1:CD%(AINDEX) =  - 1:OLO%(AINDE
X) = 83: IF ADR = B THEN ADR = 83:AINDEX = AINDEX - 1:CD%(AI
NDEX) =  - 1
430  IF RB = 21 THEN BINDEX = AINDEX - 1: IF BINDEX <  = 0 T
HEN AINDEX = 1:BINDEX = 1:PO% = 0: GOTO 310
440  IF RB = 21 THEN ADR = OLO%(BINDEX): IF ADR =  - 1 THEN
BINDEX = BINDEX - 1: GOTO 440
450  IF RB = 21 THEN ADR = OLO%(BINDEX):SH = ADR:FLAG = INDP
%(BINDEX - 1): IF FLAG = 2 OR FLAG =  - 1 THEN FLAG = INDP%(
BINDEX - 2): IF FLAG = 2 THEN FLAG = INDP%(BINDEX - 3)
460  IF RB = 21 THEN 310
470  IF RB = 27 THEN  GOSUB 270: RETURN
480  IF RB = 13 THEN CD%(AINDEX) = RB: IF PO% = 1 THEN PO% =
 0: GOTO 370
490  IF RB = 13 THEN  GOSUB 270: RETURN
500  REM == 2-SHAPE CHARS HAVE ASCI(72,90) ===
510  REM
520 ADR = F1%(RB):SH = ADR: IF RB >  = 8 AND RB <  = 18 THEN
 INDP%(AINDEX) = 2:FLAG = 1: GOTO 560
530  IF RB = 0 OR RB = 3 OR (RB >  = 19 AND RB <  = 71) THEN
 INDP%(AINDEX) = 1:FLAG = 1: GOTO 560
540 INDP%(AINDEX) = 0: IF RB >  = 72 AND RB <  = 90 THEN SET
$ = "2":FLAG = 0: GOTO 560
550  IF RB >  = 1 AND RB <  = 6 THEN SET$ = "4": IF FLAG = 0
 THEN SH = SH + 2:ADR = SH
560  GOSUB 270
570  GOSUB 720: IF LL% = 1 THEN LL% = 0: RETURN
580  GOSUB 230:OLO%(AINDEX) = ADR:BINDEX = AINDEX:AINDEX = A
INDEX + 1: IF SET$ = "" THEN 310
590  IF QB% = 1 THEN QB% = 0:AINDEX = T: GOTO 610: REM    QB
%=             1              IF CORRECTION              OCCURS

600  GET C$:ASKC =  ASC (C$):RC = R%(ASKC): IF RC <  > 21 TH
EN CD%(AINDEX) = RC
610 FLAG = INDP%(BINDEX): IF FLAG =  - 1 THEN FLAG = INDP%(B
INDEX - 1)
620  IF RC = 65 AND RB = 87 THEN  GOSUB 270: GOSUB 770:ADR =
 B: GOSUB 720: GOSUB 230:SH = B:SET$ = "":FLAG = 1:OLO%(BIND
EX) = SH:INDP%(BINDEX) = 1:INDP%(AINDEX) =  - 1:OLO%(AINDEX)
 =  - 1
630  IF RC = 65 AND RB = 87 AND AINDEX - BINDEX >  = 2 THEN
CD%(AINDEX) = CD%(AINDEX - 1):CD%(AINDEX - 1) = 65:SH = OLO%
(AINDEX - 1):ADR = SH:OLO%(AINDEX) = OLO%(AINDEX - 1):OLO%(A
INDEX - 1) =  - 1:INDP%(AINDEX) = INDP%(AINDEX - 1):INDP%(AI
NDEX - 1) =  - 1
640  IF RC = 65 AND RB = 87 THEN AINDEX = AINDEX + 1: GOTO 3
10
650  IF RC < 8 OR (RC > 18 AND RC < 32) OR (RC > 57) THEN SE
T$ = "":RB = RC: GOTO 420
660  IF RC >  = 8 AND RC <  = 18 AND  NOT (RC = 13) THEN P =
 1:ADR = F1%(RC):OLO%(AINDEX) = ADR:INDP%(AINDEX) = 2:AINDEX
 = AINDEX + 1: GOSUB 720: GOTO 600
670  REM  --- IF C$ IS A DELIMITER ---
680  IF SET$ = "2" OR (SET$ = "4" AND FLAG = 1) THEN  GOSUB
270: GOSUB 770:SH = SH + 1:ADR = SH:OLO%(BINDEX) = ADR: GOSU
B 720: GOTO 700
690  IF SET$ = "4" AND FLAG = 0 THEN  GOSUB 270: GOSUB 770:S
```

```
H = SH + 1:ADR = SH:OLO%(BINDEX) = ADR: GOSUB 720
700 SET$ = "":ADR = F1%(RC):INDF%(AINDEX) = 1:FLAG = 1: IF R
C = 13 THEN RB = 13: GOTO 490
710  GOTO 570
720  REM  === SUB FOR DRAWING CHARS ===
730  IF X < MS AND SZ%(ADR) > X THEN LL% = 1: GOTO 760
740  DRAW ADR AT X,Y
750 X = X - SZ%(ADR): IF X < 8 THEN LL% = 1
760  RETURN
770  REM  === SUB FOR XDRAWING CHARS ===
780  IF X >  = X0 THEN X = X0: GOTO 820
790 X = X + SZ%(SH)
800  IF P = 1 THEN ADR = SH:P = 0
810  XDRAW ADR AT X,Y
820  RETURN
830  REM  ======= PROGRAM ENDS ========
```

Fig. 1. Source listings for FLINE-EDITOR.

IMPLEMENTING THE SYSTEM

To implement the system it was necessary to build an Arabic character generator, and that was one of the major problems facing this system. The character generator on a CRT, especially on a microcomputer, is stored either in a Read Only Memory or in Random Access Memory. In addition, it is possible to use the graphic capabilities of a microcomputer equipped with such capabilities for the same purpose (see Poole *et al.* 1981). The designers of many new microcomputers with graphic capabilities incorporated the concept of shape tables in order to supply graphics users with some powerful software for dealing with gaming figures. This facility proved to be very useful in generating Arabic characters.

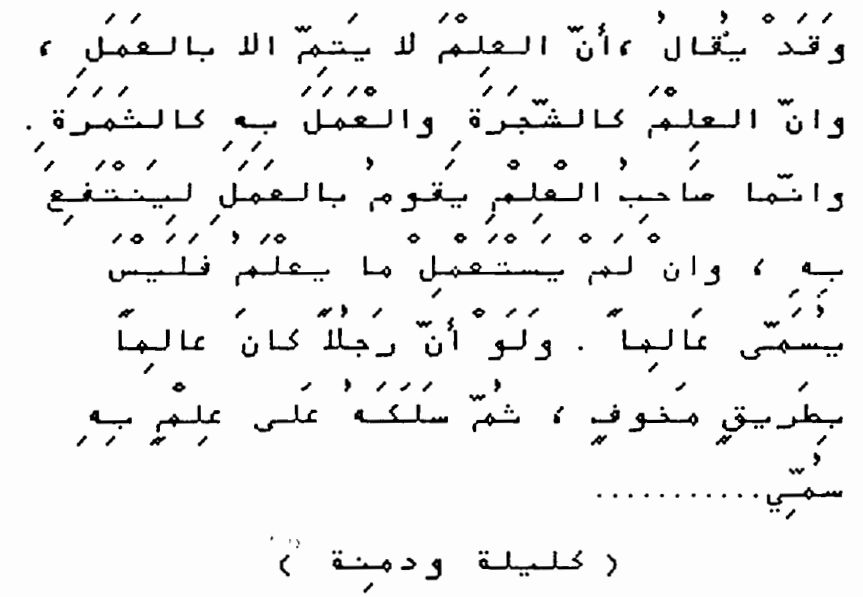

Fig. 2. Sample output of Arabic generated on a screen by FLINE-EDITOR and dumped to a dot matrix printer.

A shape table is created by outlining a shape with tiny unit vectors which are all of the same length but may be in any one of the four directions (up, down, left, right). The vectors are placed head to tail until the shape is outlined. Using a simple key, these direction vectors are converted to a string of bytes and then stored in memory as part of a shape table. One may store up to 255 shapes in a single table in memory. Once the object is represented in a shape table one can recall it using any available programming language. For example in Soft Basic, in order to draw a shape at the X,Y position on the screen we use

DRAW N AT X,Y

where N is the shape index number.

The actual procedure for creating a shape for a shape table is fairly simple but rather time consuming. A one to one correspondence was established between the available Latin keyboard and the Arabic letters (Table 1).

Fig. 1 is a listing of the program used to load the shape table into the computer memory and to implement the system. The program is a simple Arabic line editor written to test the practicality of implementing the suggested system only. A graphic program was used to output the script generated by our system on a CRT to a graphic printer (Fig. 2).

ACKNOWLEDGEMENTS

This work was carried out at the University of Louisville during a Sabbatical year from the University of Kuwait. I am grateful to Dr A. Riehl, the Head of the Department of Applied Mathematics & Computer Science at Louisville, for making my stay fruitful.

REFERENCES

Al-Banna, S. 1980. Informatics and the Arabic language. Third Arab Summer School of Science and Technology, Bloudan, Syria.

Hyder, S. 1972. A system for generating Urdu Script. Information Processing, North-Holland, Amsterdam.

Poole, L., McNiff, M. & Cook, S. 1981. Apple II User's Guide, Osborn. McGraw-Hill.

Stanat, D.F. & McAllister, D.F. 1971. Discrete mathematics in computer science. Prentice-Hall.

(Received 1 February 1983, revised 4 February 1984)

نموذج رياضي لتكوين كلمات اللغة العربية

فاروق عبد موسى
قسم الرياضيات بجامعة الكويت

خلاصة

يقدم هذا البحث دالة رياضية لتكوين مفردات اللغة العربية ويبين كيف يمكن برمجة هذه الدالة على حاسب صغرى بحيث تؤدي إلى اختصار عدد المفاتيح المستخدمة لادخال المعلومات في لوحة مفاتيح الأحرف العربية على أجهزة الاتصال الطرفية مع الحاسبات الالكترونية .